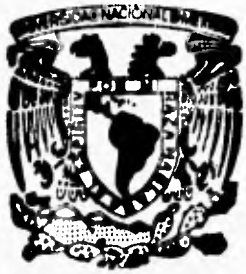


UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

DISEÑO Y CONSTRUCCION DEL MODULO DE CONTROL DE UNA SONDA ROBOT

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO MECANICO ELECTRICISTA
P R E S E N T A :

RENE TOLEDO HUITRON

ING. ROMAN V. OSORIO COMPARAN

CD. UNIVERSITARIA, 1995

INDICE

CAPITULO 1

OBJETIVOS e INTRODUCCIÓN

1. OBJETIVOS e INTRODUCCIÓN

OBJETIVOS:

- Diseño y Construcción del Control de Movimiento de una Sonda Robot durante su Trayectoria.

- Adquisición de los Parámetros Críticos de una Planta Nuclear.

- Realización de Rutinas de Muestreo por medio de un Manipulador de Seis Grados de Libertad.

INTRODUCCIÓN.

La robótica es una tecnología con un futuro y también es una tecnología para el futuro. Si continúan las tendencias actuales, y si algunos de los estudios de investigación en laboratorio actualmente en curso se convierten finalmente en una tecnología factible, los robots del futuro serán unidades móviles que tendrán la capacidad de desplazarse en cualquier tipo de condiciones de terreno, con más de un brazo, capacidades de sensores múltiples con la misma velocidad de procesamiento de datos y de cálculo que las grandes computadoras actuales. Podrán responder a órdenes dadas con voz humana. Asimismo poseerán la habilidad de recibir instrucciones generales y traducirlas, utilizando inteligencia artificial, en un conjunto específico de acciones requeridas para llevarlas a cabo. Podrán ver, oír, palpar, aplicar una fuerza media con precisión a un objeto y desplazarse por sus propios medios no sólo con la ayuda de ruedas, sino con extremidades.

Los robots del futuro se diseñarán para ser mecánicamente más eficientes y utilizarán sistema de potencia y actuación mejoradas cuando se les compara a los robots actuales. El diseño del robot es probable también que sea modularizado, de forma que se puedan construir robots para objetivos diferentes a partir de componentes que son relativamente estándares con considerables ahorros de energía.

En resumen, los futuros robots tendrán muchos de los atributos de los seres humanos. El paso del presente al futuro exigirá mucho trabajo de ingeniería electrónica, ingeniería mecánica, informática, ingeniería industrial, tecnología de materiales, ingeniería de sistema de fabricación y un amplio estudio de las ciencias sociales, a fin de entender y minimizar el rechazo de los robots por parte de los trabajadores de las industrias.

Como se mencionó anteriormente es necesario continuar con las investigaciones para mejorar los desarrollos actuales, pero de ninguna manera estamos limitados, pues en el campo de la Ingeniería Electrónica contamos actualmente con una amplia variedad de dispositivos electrónicos, con los cuales es posible realizar desarrollos que puedan contribuir al avance de la Robótica. Entre estos dispositivos, los que sobresalen son los microprocesadores y los microcontroladores. Lo anterior se puede afirmar, puesto que si se observa el circuito electrónico de alguna maquinaría de alta tecnología o incluso de un Robot Industrial se encontrará uno de estos dispositivos, que debido a sus características puede ser usado como un sistema de adquisición de datos, en aplicaciones de control, tales como locomoción, navegación o control directo de brazos mecánicos. Con este circuito, un cambiador de nivel y un programa relativamente sencillo en lenguaje de alto nivel es posible interactuar con una computadora personal a través de su puerto de comunicación serie, con lo que se pueden obtener ventajas adicionales que ofrece la computadora e incluso tener comunicación a distancia para realizar el control y la adquisición de la información en tareas que representen un alto riesgo para el hombre.

El nombre del proyecto que es desarrollado es "Sonda Robot", el cual será utilizado en una instalación radioactiva; tiene el propósito de desarrollar actividades de monitoreo e inspección en una planta nuclear, es decir contará con un sistema de tracción por medio del cual se posicionará la Sonda en el lugar deseado en forma completamente autónoma, permitiendo que un solo operario pueda manipularlo a distancia por medio de un Control Remoto, con el fin de llevarlo hasta la zona de alto riesgo deseada, esto es en áreas de alta rapidez de exposición, en donde podrá realizar la adquisición de datos, como son las condiciones radioactivas ambientales (Radiación Gama y Radiación Beta), así como, las condiciones atmosféricas (temperatura y humedad relativa), con lo que se evita la necesidad de exponer al personal de operación a una posible radiación, además de que se podrá tener un monitoreo permanente de los parámetros críticos de la planta puesto que se integrará un sistema de transmisión de datos (el cual forma parte de otro desarrollo que se está realizando) con lo que la información obtenida podrá ser transmitida en ese momento a una Computadora.

CAPITULO 2

GENERALIDADES

En éste capítulo se presenta la descripción detallada de los elementos requeridos para llevar a cabo el desarrollo y construcción de la Sonda Robot.

2. GENERALIDADES

2.1 MOTORES DE PASOS

El motor de pasos es una forma de motor de CA que está diseñado para girar un determinado número de grados por cada pulso eléctrico que se aplique a su unidad de control. Los tamaños de los pasos pueden ir desde menos de un grado hasta 45 grados o más. El motor de pasos se usa a menudo en sistemas de control digital, en los cuales el motor recibe órdenes de lazo abierto en forma de un tren de pulsos para hacer girar su eje o mover un objeto una determinada distancia.

Una ventaja notable del motor de pasos es su compatibilidad con los sistemas electrónicos digitales. Estos sistemas son cada vez más comunes en una amplia gama de aplicaciones y al mismo tiempo se están fabricando con mayor potencia y menor costo. Las aplicaciones típicas son, entre otras, motores de alimentación de papel en máquinas de escribir e impresoras, posicionamiento de cabezas de impresión o plumas en graficadores XY, cabezas de grabación en impulsores de discos de computadora, y posicionamiento de mesa de trabajo y de herramienta en equipos de maquinado controlados numéricamente. En muchas aplicaciones se puede obtener la información sobre la posición tan solo con mantener una cuenta de los pulsos que se envían al rotor, y no son necesarios sensores de posición ni control por retroalimentación.

Distinto a los motores convencionales, los cuales se mueven continuamente, un motor de pasos se mueve en distintos intervalos. Este intervalo de acción es muy usado para muchas aplicaciones como ya se mencionó. Los motores de pasos permiten tener un control de posición, velocidad, distancia y dirección. Porque cada paso que ejecuta el motor mueve la flecha con un ángulo constante, el único error que se presenta es en el último paso, y este error es generalmente del 5% del ángulo de giro. (Con el motor alimentado, el error disminuye considerablemente).

Los motores de pasos se fabrican en una amplia variedad de diseños y configuraciones. Estas últimas comprenden las de reluctancia variable, imán permanente e híbridas. La resolución angular del motor de reluctancia variable se determina por el número de dientes del estator, la cuál se puede aumentar mediante técnicas como el dentado.

De los tres tipos de motores de pasos los más utilizados en el marcado mundial son los de reluctancia variable y los híbridos. En la práctica se ha visto que los de reluctancia variable son atractivos para pasos angulares grandes (por ejemplo: 15, 30 o 45 grados), mientras que los híbridos pueden ser mejor adaptados cuando pequeños ángulos son requeridos (por ejemplo: 1.8 ó 2.5 grados).

Motor de reluctancia variable.

Las configuraciones de motores de reluctancia variable constan de un rotor y un estator únicos con fases múltiples. A un motor de pasos con esta configuración se le llama de pasos de reluctancia variable y conjunto único. Una forma alterna del motor de pasos de reluctancia variable se llama motor de pasos de reluctancia variable y varios conjuntos. Se puede considerar que el motor, en esta configuración, está constituido por un conjunto de motores monofásicos de reluctancia variable desplazados axialmente y montados en el mismo eje.

Un diagrama simplificado de un motor de reluctancia variable de 30^{0} por paso es mostrado en la Figura 2.1. El estator es fabricado con una pila de laminas de acero, y presenta seis polos igualmente proyectados y espaciados, o dientes, cada uno con una bobina simple separada. El rotor, el cual puede ser sólido o laminado, tiene cuatro dientes proyectados, del mismo ancho de los dientes del estator. Existe un pequeño hueco - típicamente entre 0.02 mm y 0.2 mm - entre el rotor y los dientes del estator. Cuando ninguna corriente está fluyendo en alguna de las bobinas del estator, el rotor por lo tanto estará completamente libre para girar. Pares de bobinas del estator opuestas diametralmente son conectadas en serie, por lo que si una de ellas actúa como un polo norte, la otra actúa como un polo sur. Hay entonces tres circuitos de estator independientes, o fases, y cada una puede ser alimentada con corriente directa proveniente del circuito de control.

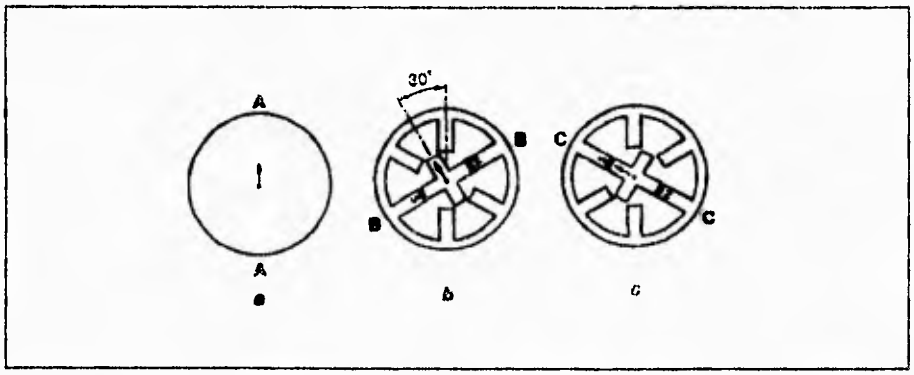


Figura 2.1. Principio de operación de un motor de pasos de reluctancia variable de 30°/paso.

Cuando una fase es energizada, un campo magnético con sus ejes a lo largo de los polos del estator de la fase A es creado. El rotor por lo tanto es atraído a una posición donde el par de polos del rotor (diametralmente opuestos) se alinean con el campo, en línea con el polo de la fase A, como se muestra en la figura 2.1a. Cuando la fase A es desactivada, y la fase B es activada, el segundo par de polos del rotor tendera a alinearse con los polos del estator de la fase B, el

rotor se moverá 30^{o} en sentido antihorario a su nueva posición, como se muestra en la Figura 2.1b. Un paso adicional en sentido antihorario de 30^{o} ocurrirá cuando la fase B es desactivada y la fase C es activada. En esta etapa el par de polos originales del rotor se ponen en movimiento nuevamente, pero esta vez son atraídos por los polos del estator C, como se muestra en la Figura 2.1c. Con reiteradas activaciones de las fases en la secuencia ABCA..., provocarán que el rotor gire con pasos de 30^{o} en sentido antihorario, mientras que si la secuencia es ACBA..., girará en sentido horario. Este modo de operación es conocido como '1-fase-encendida', y es la manera más simple de poner en operación al motor. Nótece que la polaridad de la corriente de excitación no es significativa. El motor se alineará en la misma forma sin importar la dirección de la corriente.

Motor de imán permanente.

En cuanto al motor de pasos de imán permanente, el rotor tiende a alinearse con una fase cuando es excitada individualmente. A diferencia del motor de reluctancia variable, la alineación del rotor en el motor de pasos de imán permanente depende de la dirección de las corrientes de fase. Si se invierten estás se hará que el rotor invierta su orientación. En el motor de pasos de imán permanente, también a diferencia de su contraparte, el motor de reluctancia variable, se generará el par que tiende a alinear al rotor con los polos del estator aun cuando no haya excitación aplicada a los devanados de fase. Así el rotor tendrá posiciones particulares de descanso sin excitación, un hecho del que se puede sacar ventaja para algunas aplicaciones.

El imán permanente del motor de pasos trabaja con las características básicas de los imanes: como son la de que polos iguales se repelen y polos opuestos se atraen. La parte giratoria del motor paso a paso consiste en imanes extendidos hacia afuera del eje. La funda (estator) del motor contiene electroimanes, los cuales interactúan con el imán permanente del rotor provocando (o impidiendo) movimiento. Existen básicamente dos tipos de motores de pasos de imanes permanentes:

- Bipolar, con dos bobinas y 4 alambres
- Unipolar, con cuatro bobinas y 8 o 6 alambres

Un motor bipolar tiene solamente un embobinado por cada imán (dos embobinados en total). Para poder manejar el electroimán con orientación Norte y Sur, la corriente debe ser invertida. Motores unipolares tienen dos embobinados por cada imán (4 embobinados en total). esto significa que cada imán tiene cuatro alambres (dos pares). Solamente una de estas bobinas debe ser activada a la vez, de otra manera puede existir un conflicto entre ellas. Algunas veces dos de los cuatro alambres son conectados juntos en el interior del motor de pasos, y así sólo 3 alambres salen para cada imán. Esto da un total de 6 alambres en vez de 8.

Motor híbrido.

El motor de pasos híbrido combina características de los motores de reluctancia variable y de imán permanente. La configuración del motor de pasos híbrido se parece mucho a la de un motor de pasos de reluctancia variable de varios conjuntos; a diferencia de éste, los conjuntos del rotor en el motor de pasos híbrido están separados por un imán permanente dirigido axialmente. Como resultado de ello se puede considerar que un extremo tiene un polo norte magnético y el otro un polo sur.

El motor de pasos híbrido difiere mucho de uno de pasos de reluctancia variable porque la estructura polar del estator es continua en la longitud del rotor.

El diseño híbrido de motor de pasos presenta ventajas sobre el de imán permanente que se describió anteriormente. Puede alcanzar con facilidad tamaños de pasos pequeños con una estructura sencilla de imán, mientras que un motor sólo de imán permanente necesitaría un imán permanente de varios polos. En comparación con el motor de pasos de reluctancia variable, el tipo híbrido puede necesitar menos excitación para lograr un par determinado debido a que una parte de la excitación es suministrada por el imán permanente. Además, el motor híbrido tiende a mantener su posición cuando se quita la excitación del estator, al igual que el de imán permanente.

Una vista seccionada del interior de un motor híbrido típico de 1.8^0 es mostrado en la Figura 2.2. El estator tiene 8 polos principales, cada uno con 5 dientes, y cada polo principal contiene una bobina simple. El rotor tiene dos conjuntos terminales, cada uno con 50 dientes, y separados por un imán permanente. Los dientes del rotor presentan la misma separación que los dientes situados en los polos del estator, y están desplazados en un ángulo igual a la mitad del paso polar del rotor; así que la línea central de un diente en su extremo coincide con la ranura del otro. El imán permanente es axialmente magnetizado, por lo que un grupo de dientes del rotor nos da una polaridad norte, y el otro una polaridad sur.

Cuando ninguna corriente está fluyendo en los devanados, la única fuente de flujo magnético a través del hueco de aire es el imán permanente. Si no hubiera desplazamiento entre los dos juegos de dientes del rotor, existiría un fuerte y periódico par de alineamiento cuando el rotor fuera girado, y todo el tiempo el grupo de dientes del estator estarían en línea con los dientes del rotor con lo que se obtendría una posición de equilibrio estable. Sin embargo existe un desplazamiento, y esto causa que el par de alineamiento debido al imán casi sea eliminado. En la práctica un pequeño par de "retención" permanece, y puede ser percibido si la flecha es girada cuando el motor es desenergizado: el motor tiende a ser mantenido en su posición por el par de retención. Esto es algunas veces muy útil: por ejemplo es habitualmente suficiente para sostener el rotor cuando está fijo, situación que se presenta cuando la fuente está apagada, así que el motor puede ser dejado de noche sin ningún temor de que se pueda mover accidentalmente a una nueva posición.

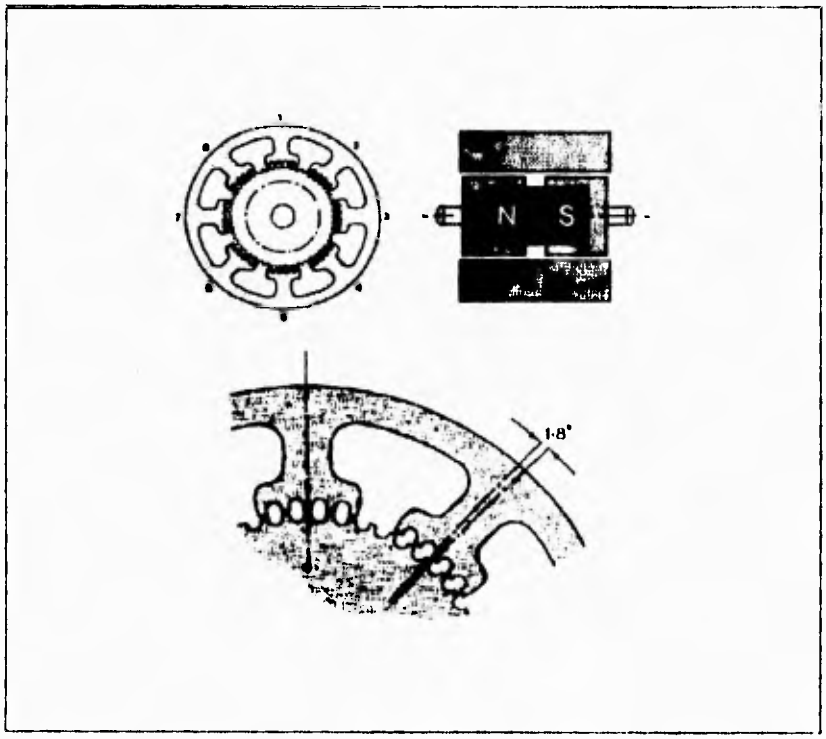


Figura 2.2. Motor de pasos híbrido (200 pasos/revolución). El esquema muestra los dientes del rotor y del estator alineados, e indica el ángulo de paso de 1.8°.

Los ocho embobinados están conectados de tal manera que forman dos fases o embobinados. Las bobinas sobre los polos 1, 3, 5 y 7 forman la fase A, mientras que aquellas sobre los polos 2, 4, 6 y 8 forman la fase B. Cuando en la fase A circula corriente positiva, los polos 1 y 5 del estator están magnetizados como sur, y los polos 3 y 7 como norte. Los dientes situados en el extremo norte del rotor son atraídos a los polos 1 y 5 mientras que los dientes desplazados en el extremo sur del rotor son atraídos y situados en línea con los dientes de los polos 3 y 7. Para lograr que el motor realice un paso, la fase A es apagada, y la fase B es energizada ya sea con corriente positiva o negativa, dependiendo del sentido de rotación requerido. Esto causará que el rotor se mueva un cuarto de la separación de los dientes (1.8^{o}) a una nueva posición de equilibrio (paso).

El motor es movido paso a paso continuamente al energizar las fases en secuencia +A, -B, -A, +B, +A (sentido horario) o +A, +B, -A, -B, +A (en sentido antihorario). De lo anterior queda claro que será requerida una fuente bipolar. Cuando el motor es operado en esta forma es referido como '2-fases, con alimentación bipolar'.

Si una fuente bipolar no está disponible, el mismo diseño de polarización de polos es logrado de una manera diferente. Cada polo lleva dos bobinas idénticas ('embobinado de dos hilos'). Para magnetizar el polo 1 al norte, un corriente positiva es alimentada en uno de los grupos de bobinas de la fase A. Pero para magnetizar el polo 1 al sur, la misma corriente positiva es alimentada al otro grupo de bobinas de la fase A, la cual tiene el sentido contrario de embobinado. En total, hay entonces cuatro embobinados, y cuando el motor es operado en esta forma es referido como '4-fases, con alimentación unipolar'. Puesto que solamente la mitad de los embobinados es usado en cualquiera de los instantes, la salida nominal hablando térmicamente es claramente reducida en comparación con la operación bipolar.

Una especificación importante de los motores de pasos es el número de grados que cambia el eje del motor cada vez que los polos son cambiados, esto es el número de grados que el eje gira por cada paso (step angle). Los ángulos más comunes son 1.8 grados y 7.5 grados, lo que corresponde a 200 y 48 pasos por revolución del eje respectivamente.

Debe hacerse notar que el objetivo de controlar un motor de pasos para obtener la respuesta deseada en condiciones dinámicas de estado transitorio es bastante compleja y esta en tema de investigación.

2.2 Semiescalonamiento.

Además de la técnica de escalonamiento simple explicada anteriormente ('1-fase-encendida'), que es la más sencilla y la más comúnmente utilizada, existen otras dos técnicas, las cuales también son frecuentemente utilizadas. Estas dos formas de operación son conocidas como '2-fases-encendidas' y 'semiescalonamiento' (half-stepping). Con 2-fases-encendidas se puede obtener un gran par de retención y una mejor respuesta amortiguada paso a paso que en el modo de 1-fase-encendida; mientras que el modo de semiescalonamiento permite que el ángulo de paso efectivo se reduzca a la mitad - por esta razón la resolución es del doble - lográndose con esto una rotación mucho más suave en la flecha.

En el modo de 2-fases-encendidas, dos fases son excitadas simultáneamente. Cuando las fases A y B son energizadas, por ejemplo, el rotor experimenta pares desde ambas fases, y llega al descanso en un punto intermedio entre las dos posiciones contiguas de los pasos completos. Si

las fases son activadas en la secuencia AB, BC, CA, AB etc, el motor realizará pasos completos (30^{o}), como en el modo de 1-fase-encendida, pero su posición de equilibrio se intercalará entre las dos posiciones de paso completo.

Para obtener el escalonamiento las fases son excitadas en la secuencia A, AB, B, BC etcétera, alternadamente en los modos 1-fase -encendida y 2-fases-encendidas. Esto es algunas veces conocido como excitación "ondulada", y causa que el rotor avance (en el caso de un motor de reluctancia variable de 30^{o}) con pasos de 15^{o}, o que es lo mismo, la mitad del ángulo de paso completo. Como podría esperarse, continuos semiescalonamientos producirán habitualmente una rotación suave de la flecha, lo que no sucede en el modo de paso a paso completo, lográndose también doblar la resolución.

El precio que se paga por el incremento del par de retención es el incremento en la disipación de potencia en los embobinados, el cuál es del doble en comparación con la operación en modo normal (1-fase-encendida), mientras que el par de retención es incrementado con un factor menor a dos, así que el par por watt es reducido.

Estrictamente, tal procedimiento es solamente válido cuando las dos fase son magnéticamente independientes, o las partes comunes del circuito magnético no están saturadas. Este no es el caso en muchos motores, en los cuales las fases comparten un circuito magnético común que opera bajo muy altas condiciones de saturación.

2.3 ESTUDIO DE LOS ROBOTS

La palabra robot proviene de la palabra checa robota, que significa trabajo. Una definición utilizada por el Robot Institue of America da una descripción más precisa de los robots industriales: un robot es un manipulador reprogramable multifuncional diseñado para mover materiales, piezas o dispositivos especializados, a través de movimientos programados variables para la realización de una diversidad de tareas. En suma, un robot es un manipulador reprogramable de uso general con sensores externos que pueden efectuar diferentes tareas. Con esta definición el robot debe poseer inteligencia que se debe normalmente a los algoritmos de computador asociados con un sistema de control y sensorial.

Está claro que hay otras máquinas especializadas que pueden realizar trabajos semejantes, pero que no son robots. La diferencia es que el robot no es una máquina especializada, sino que puede ser parte de una línea de producción automática flexible o en nuestro caso es un autómata que puede efectuar múltiples tareas previamente programadas o también, tener la capacidad de seguir órdenes. Como el control del Robot se realiza por medio de un microcontrolador sus posibilidades llegan a ser prácticamente ilimitadas.

Existen alrededor de 250 fabricantes de robots en los Estados Unidos, Europa y Japón, haciendo muy difícil la identificación de todas las partes usadas en los robots disponibles. Sin embargo, hay componentes comunes que pueden ser examinados para una mejor perspectiva en como trabaja un robot.

El manipulador es una de las tres partes básicas del robot. Las otras dos son el controlador y la fuente de energía.

Las dos propiedades que caracterizan a un robot son: la versatilidad y la autoadaptabilidad al entorno.

a) La versatilidad: es la potencialidad (posibilidad) estructural (mecánica) de ejecutar tareas diversificadas y/o ejecutar una misma tarea de forma diversificada. Esto impone al robot una estructura mecánica de geometría variable. Todos los robots tienen esta propiedad.

b) La autoadaptabilidad al entorno: esto significa simplemente, que un robot debe, por sí solo, alcanzar su objetivo (la ejecución de una tarea), a pesar de las perturbaciones imprevistas (pero limitadas) del entorno, a lo largo de la ejecución de la tarea. Esta propiedad que empieza justamente a aparecer en los robots industriales, supone que el robot sea consciente de su entorno, por lo tanto que posea sentidos artificiales. En este campo, lo que se ha desarrollado todavía es muy modesto con relación a la potencia de comprensión del entorno que tiene el hombre. Pero es por este lado donde las investigaciones son más intensas.

Un robot operacional puede representarse por cuatro entidades unidas entre sí:

1) el sistema mecánico articulado dotado de sus motores (que se llaman también accionadores o actuadores). Estos pueden ser eléctricos, hidráulicos o neumáticos. Arrastran a las articulaciones del robot, mediante las transmisiones, también de diversa naturaleza (cables, cintas, correas con muescas, engranajes, tornillos sin fin, etc.).

2) el entorno: es el universo en el que está sumergida la primera entidad. Para los robots con puesto fijo, se reduce a lo que se encuentra en el espacio alcanzable del robot, definido por el volumen barrido cuando éste pasa por todas las configuraciones posibles. En este entorno el robot encontrará obstáculos que debe evitar y objetos de interés, es decir, aquellos sobre los que debe actuar. En consecuencia, existe una interacción entre la primera entidad (el robot << físico >>) y el entorno.

3) tareas a realizar: es el trabajo que se desea que haga el robot de la primera entidad. Es preciso por lo tanto poder describirlo. Esto se hace mediante lenguajes que pueden ser por gestos (se enseña al robot lo que debe hacer), orales (se le habla) o por escrito (se le escribe en un lenguaje que comprenda).

4) el "cerebro" del robot: es el órgano de tratamiento de la información. Para los robots menos evolucionados casi siempre es un autómata programable. Para los robots avanzados es un miniordenador numérico o un microcontrolador.

Como para el hombre, el "cerebro" tiene el papel principal. Teniendo almacenada la siguiente información:

a) un modelo del robot físico, es decir, las relaciones entre las señales de excitación de los accionadores y los desplazamientos que son consecuencias de ellas.

b) un modelo del entorno, que es la descripción de lo que se encuentra en el espacio que puede alcanzar, por ejemplo, las zonas que no debe atravesar ya que hay obstáculos.

c) programas que le permitan comprender las tareas que se le pide que realice.

d) programas que le permitan controlar el robot físico con el fin de que este ejecute lo que debe. Son los algoritmos de control.

A lo largo de la ejecución de una tarea, el ordenador, en todo instante:

- percibe el estado del robot gracias a la información propioceptiva, que es la información que nos da la configuración de las articulaciones o incluso el estado del robot, con relación a un valor de referencia llamado configuración de inicialización; esta información es proporcionada por sensores ópticos, potenciómetro, etc.

- percibe el estado del entorno gracias a la información extereoceptiva, que es la información que nos permite saber de la existencia de objetos alrededor del robot, esto se realiza por medio de cámaras, detectores de fuerza, sensores de proximidad, sensores táctiles, etc.

- recurre a diversos modelos y programas registrados.

- genera una orden (es decir, señales de potencia de los actuadores), que hace progresar al robot físico hacia la ejecución correcta de la tarea que se le ha pedido.

Hablando de los costos de producción, desde su aparición hace algunos años, hasta nuestros días, se han logrados reducir considerablemente, debido a que los constantes desarrollos en microelectrónica, como son el caso de los microprocesadores y más recientemente de los microcontroladores, permiten que la utilización industrial de los robots sea posible puesto que el costo de la parte electrónica con respecto a la mecánica es mucho menor, lo que permite la utilización de robots en la industria por ser más accesibles económicamente hablando.

El campo de aplicación de la robótica influye profundamente en la forma y propiedades de las máquinas y robots.

Puede facilitarse la comprensión considerando tres grandes campos de aplicación. En el interior de cada campo, los problemas a resolver tienen cierto parecido. De un campo a otro son bastante diferentes.

1. **El campo de la producción**: concretamente aquí es donde los industriales han desarrollado el máximo esfuerzo, ya que en la utilización de los robots ven numerosas ventajas, en primer lugar, la disminución de la mano de obra.

De hecho, la asociación de robots entre sí y con otras máquinas, aporta dos ventajas fundamentales con relación a los modos de producción tradicionales:

a) la automatización casi integral de la producción que puede acompañarse:
   1) de una mejor calidad del producto terminado.
   2) de una mayor fiabilidad en el mantenimiento de esta calidad.
   3) de una mejor adaptación de la cantidad producida a la demanda.

b) la rapidez de reconfiguración de la unidad de producción cuando se pasa de la fabricación de un producto a la de un producto similar, o bien, cuando un incidente inmoviliza una máquina de la unidad de producción.

2. El campo de la exploración: aquí se trata de un problema diferente. Se requieren ejecutar trabajos en un lugar al que el hombre no puede ir porque el medio es hostil. Es el caso para:

- el medio submarino
- el medio espacial
- el medio irradiado de las centrales nucleares

pero también para intervenir en un incendio, etc.
La robótica permite ensayar dos tipos de solución para estas intervenciones:

a) el robot autónomo: es el que se va a enviar al medio hostil, programándole su misión; por ejemplo: recoger ciertas piedras en el planeta Marte, o bien inspeccionar las soldaduras de una central nuclear.

Hoy en día saben hacerse robots autónomos, pero el trabajo que puede encomendárseles debe ser de una gran simplicidad y sobre todo no puede exigirse que el robot tenga que "reflexionar", es decir, comprender su entorno. Por eso, la solución operacional, aun siendo muy mejorable es la de:

b) la teleoperación: (llamada también telepresencia), la cual consiste en enviar una máquina (un robot que se llama máquina esclava), al medio hostil y poder controlarla a distancia desde un punto llamado puesto maestro, al mando del cual se encuentra un operador.

Es pues el operador quien efectúa todas las tareas de reflexión y de activación de los movimientos de la máquina esclava. Eso requiere, evidentemente, el envío al puesto maestro de lo que sucede en el universo esclavo, ya sea mediante la adquisición de datos tales como posición, sensores de proximidad o incluso presencia de cámaras.

3. El campo de la asistencia individual: existe un campo donde se desarrolla la robótica de asistencia individual y es el de la robótica médica, que permite mejorar las condiciones de vida de los paralíticos (parapléjicos y tetrapléjicos) o de los amputados.

La robótica cubre entonces el campo de las:

- prótesis (manos y piernas por ejemplo).
- órtesis: son estructuras rígidas motorizadas, que se ponen alrededor del miembro paralizado y lo arrastran en sus movimientos.
- telétesis: están destinadas a los paralíticos de los cuatro miembros y son robots que el afectado controla a distancia (mediante teleoperación), a partir de las zonas de motricidad voluntaria que haya podido conservar (por ejemplo la lengua, la boca, los músculos de los ojos, etc.).

2.4 Manipuladores.

Mecánicamente, un robot se compone de una base, un cuerpo (cintura y/u hombro), un brazo y una muñeca, (en ocasiones un antebrazo), que todos en conjunto se denomina el Manipulador; haciendo notar que en general se habla de manipulador cuando el brazo o elemento articulado no posee sensores que interactúen con el medio, mientras que se habla de robot si el manipulador o brazo está provisto de sensores que interactúan con el medio. Unida a la muñeca del manipulador va una mano. El nombre técnico aplicado a la mano es "efector final", el cual no se considera como parte de la anatomía del manipulador.

Los manipuladores industriales están diseñados para realizar un trabajo productivo. El trabajo se realiza permitiendo que el robot desplace su cuerpo, brazo y muñeca mediante una serie de movimientos y posiciones. Unido a la muñeca como ya se mencionó está el efector final, que es utilizado por el manipulador para efectuar una tarea específica. Los movimientos del robot pueden dividirse en dos categorías generales: movimientos de brazo y cuerpo y movimientos de la muñeca. Los movimientos de articulaciones individuales asociados con estas dos categorías se denomina, a veces, por el término de "grado de libertad", que no son otra cosa que los movimientos independientes disponibles en el volumen de trabajo del manipulador, el cual se refiere al espacio dentro del cual el manipulador puede mover el extremo de su muñeca.

El convenio de utilizar el extremo de la muñeca para definir el volumen de trabajo del manipulador se adopta para evitar la complicación de diferentes tamaños de efectores finales, que podrían unirse a la muñeca del manipulador. El efector final es una adición al manipulador básico y no debe contarse como parte del espacio de trabajo del manipulador.

El volumen de trabajo viene determinado por las siguientes características físicas del manipulador:

La configuración física del manipulador.
Los tamaños de los componentes del manipulador.
Los límites de los movimientos de las articulaciones del robot.

Los manipuladores industriales están disponibles en una amplia gama de tamaños y configuraciones físicas. La gran mayoría de los manipuladores comercialmente disponibles en la actualidad tienen una de estas cuatro configuraciones básicas.

1. Configuración polar o esférica (un eje lineal y dos ejes rotacionales).
2. Configuración cilíndrica (dos ejes lineales y un eje rotacional).
3. Configuración de coordenadas cartesianas (tres ejes lineales).
4. Configuración de brazo articulado o de revolución (tres ejes rotacionales).

La configuración polar (Figura 2.3d), utiliza un brazo telescópico que puede elevarse o bajar alrededor de un pivote horizontal. Este pivote está montado sobre una base giratoria. Estas diversas articulaciones proporcionan al manipulador la capacidad para desplazar su brazo dentro de un espacio esférico, y de aquí la denominación de manipulador de "coordenadas esféricas" que se suele aplicar a este tipo. Entre otros robots que presentan está configuración esta el Unimate serie 2000 y el Maker 110.

La configuración cilíndrica (Figura 2.3c), utiliza una columna vertical y un dispositivo de deslizamiento que puede moverse hacia arriba o hacia abajo a lo largo de la columna. El brazo está unido al dispositivo deslizante de modo que puede moverse en sentido radial con respecto a la columna. Haciendo girar la columna, el robot es capaz de conseguir un espacio de trabajo que se aproxima a un cilindro.

El manipulador de coordenadas cartesianas (Figura 2.3a), utiliza tres dispositivos deslizantes perpendiculares para construir los ejes *x*, *y* y *z*. Desplazando los tres dispositivos deslizantes entre sí, el manipulador es capaz de operar dentro de una envolvente rectangular de trabajo. Un ejemplo de esta configuración es el robot IBM RS-1 (denominado Modelo 7565).

El manipulador de brazo articulado (Figura 2.3b), tiene una configuración similar a la del brazo humano. Está constituido por dos componentes rectos, que corresponden al antebrazo y al brazo humano, montados sobre un pedestal vertical. Estos componentes están conectados por dos articulaciones giratorias que corresponden al hombro y al codo. Una muñeca está unida al extremo del antebrazo, con lo que se proporcionan varias articulaciones suplementarias. Entre los robots disponibles que tienen esta configuración se encuentra el Cincinnati Milacron T3 (Modelo 776). Una versión especial del manipulador de brazo articulado se presenta en el SCARA, cuyas articulaciones de hombro y de codo giran alrededor de ejes verticales, lo que proporciona una importante rigidez para el manipulador en la dirección vertical, pero una elasticidad en el plano horizontal, lo que lo hace ideal para muchas tareas de montaje.

El manipulador generalmente se puede mover con tres grados de libertad. La combinación de los movimientos posiciona la muñeca sobre la pieza de trabajo. La muñeca normalmente consta de tres movimientos giratorios. La combinación de estos movimientos orienta a la pieza de acuerdo a la configuración del objeto para facilitar su recogida. Estos tres últimos movimientos se denominan a menudo elevación (pitch), desviación (yaw) y giro (roll). Por tanto para un manipulador con seis articulaciones, las articulaciones del cuerpo, brazo y antebrazo del manipulador se emplean para situar el efector final y las articulaciones de la muñeca del manipulador se utilizan para orientar dicho efector final.

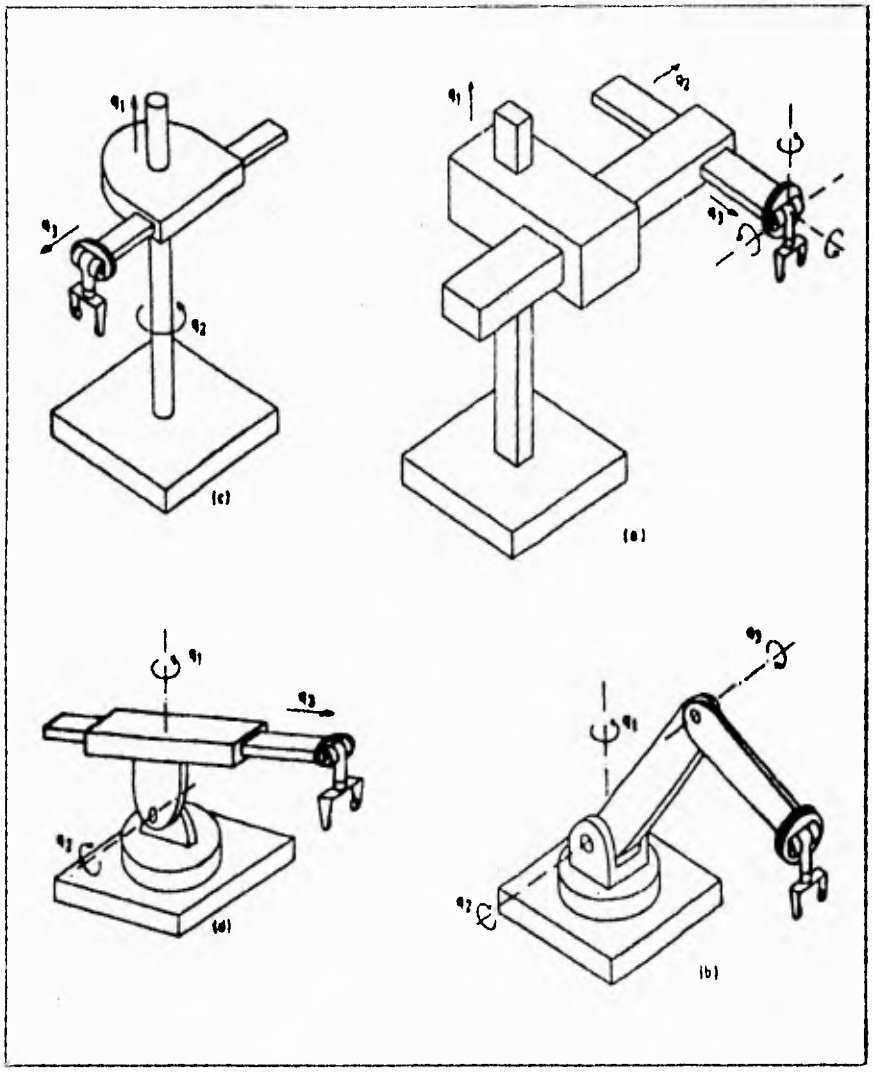


Figura 2.3. Configuraciones básicas de los manipuladores comerciales.

TIPOS DE EFECTORES FINALES

Existe una amplia gama de efectores finales para realizar una gran variedad de funciones de trabajo diferentes. Estos tipos se pueden dividir en dos categorías principales:

1. Pinzas.
2. Herramientas.

Las pinzas son efectores finales que se utilizan para agarrar y sostener objetos utilizando dedos mecánicos impulsados por un mecanismo para efectuar esta tarea. Los dedos, algunas veces llamados uñas, son los accesorios de la pinza que están en contacto con los objetos. Además, los dedos están unidos al mecanismo o son una parte integral del mismo. Si los dedos son del tipo enganche, se pueden separar y sustituir. En la mayoría de las aplicaciones dos dedos son suficientes para sostener los objetos. Las pinzas con tres o más dedos son menos frecuentes. Los objetos suelen ser piezas que tienen que ser movidas por el manipulador. Se tiende a pensar en las pinzas como en dispositivos de agarre mecánico, pero existen modos alternativos de sujeción de objetos que implican el uso de imanes, ventosas, etc.

La función del mecanismo de pinza es trasladar algo a partir de un suministro de energía que origina una acción de agarre de los dedos sobre la pieza. La energía de entrada es suministrada por el manipulador y puede ser neumática, eléctrica, mecánica o hidráulica. El mecanismo debe ser capaz de abrir y cerrar los dedos y de aplicar la fuerza suficiente contra la pieza para sostenerlo de forma segura cuando se cierre la pinza.

Existen dos formas de sostener al objeto dentro de la pinza. La primera es comprimiendo la pieza hasta alguna posición, limitando el movimiento de la pieza. La segunda forma de sujeción de la pieza es mediante el rozamiento entre los dedos y la pieza. Con este método los dedos deben aplicar una fuerza que proporcione un rozamiento suficiente para retener la pieza en contra de la gravedad, la aceleración y otras fuerzas que puedan surgir durante el tiempo de sujeción del ciclo de trabajo. Los dedos o cojinetes unidos a los dedos que hacen contacto con la pieza suelen ser fabricados de un material que es relativamente blando. Este se encarga de aumentar el coeficiente de rozamiento entre la pieza y la superficie de contacto de los dedos.

Además de las pinzas mecánicas, hay una diversidad de otros dispositivos que pueden diseñarse para elevar y sujetar objetos, como ya se mencionó; entre estos otros tipos de pinzas están incluidos los siguientes:

1. Ventosas.
2. Pinzas magnéticas.
3. Pinzas adhesivas.
4. Ganchos, cucharas y otros dispositivos diversos.

Ventosas

Las ventosas, también denominadas casquetes de vacío. Los requisitos habituales exigidos a los objetos a manipular con este tipo de pinzas son que sean planos, suaves y limpios, que son las condiciones necesarias para formar un vacío satisfactorio entre el objeto y la ventosa.

Las ventosas utilizadas en este tipo de pinzas suelen ser de material elástico, tal como caucho o plástico blando. Una excepción sería cuando el objeto a manipular esté constituido por un material blando. En este caso, la ventosa sería de una sustancia dura. La forma de la ventosa es normalmente redonda. Se requiere algún medio de extracción del aire entre la ventosa y la superficie de la pieza para crear el vacío. La bomba de vació y el tubo Venturi son dos dispositivos comunes utilizados para este propósito.

Algunas de las características y ventajas que caracterizan a la operación de las pinzas de ventosas utilizadas en las aplicaciones de robótica son:

Sólo requiere una superficie de la pieza para agarre.
Aplica una presión de distribución uniforme sobre la superficie de la pieza.
La pinza tiene un peso relativamente pequeño.
Es aplicable a una diversidad de materiales diferentes.

Pinzas magnéticas

Las pinzas magnéticas pueden ser un medio muy factible de manipular materiales ferrosos. Por lo general, las pinzas magnéticas ofrecen las siguientes ventajas en aplicaciones de manipulación robótica:

Los tiempos de captación son muy pequeños.
Pueden tolerarse variaciones en el tamaño de la pieza. La pinza no tiene que diseñarse para una pieza de trabajo particular.
Tienen la capacidad para manipular piezas mecánicas con agujeros (lo que no es posible con las pinzas de vacío).
Sólo requieren una superficie para agarre.

Los inconvenientes de las pinzas magnéticas incluyen el magnetismo residual remanente en la pinza de trabajo, que puede producir anomalías en la posterior manipulación y el posible deslizamiento lateral y otros errores, que limitan la precisión de este medio de manipulación. Otro inconveniente potencial es el problema de captar solamente una lámina a partir de una pila. La

atracción magnética tiende a penetrar más allá de la lámina superior en la pila, lo que da lugar a la posibilidad de que más de una lámina sea elevada por el imán, problema que es posible superar con ciertas técnicas.

Las pinzas magnéticas pueden dividirse en dos categorías: las que utilizan electroimanes y las que emplean imanes permanentes. Las pinzas electromagnéticas son fáciles de controlar, pero requieren una fuente de alimentación de corriente continua y una unidad de control adecuada.

Los imanes permanentes tienen la ventaja de no requerir una fuente de alimentación exterior para accionar el imán. Sin embargo, existe una pérdida de control que acompaña a esta ventaja aparente, la cual se presenta en el momento en el que ha de liberarse la pieza, pues debe proporcionarse algún medio para separar la pieza del imán. El dispositivo utilizado se denomina eyector o despegador, teniendo la función de separar mecánicamente la pieza del imán.

Pinzas adhesivas

Diseños de pinzas en los que una sustancia adhesiva realiza la acción de agarre pueden utilizarse para manipular tejidos y otros materiales livianos. Los requisitos para los elementos a manipular son que deben agarrarse por un lado solamente y que no son adecuadas otras formas de agarre. Una de las limitaciones potenciales de una pinza adhesiva es que la sustancia adhesiva pierde su pegajosidad con un empleo repetido. En consecuencia, su fiabilidad como dispositivo de agarre disminuye con cada ciclo de operación sucesivo. Para superar esta limitación el material adhesivo se carga en la forma de una cinta continua en un mecanismo de alimentación que está unido a la muñeca del manipulador. El mecanismo de alimentación funciona de manera similar a un mecanismo de la cinta de una máquina de escribir.

Ganchos, cucharas y otros dispositivos diversos

Una diversidad de otros dispositivos puede utilizarse para agarrar piezas o materiales en las aplioaciones robóticas. Los ganchos pueden emplearse como efectores finales para manipular contenedores de piezas y para cargar y descargar piezas que cuelguen de transportadores aéreos.

Las cucharas y los calderos pueden utilizarse para manipular algunos materiales en forma de polvo o líquidos. Una de sus limitaciones es que la cantidad de material recogido por el manipulador es, a veces, difícil de controlar. El derrame durante el ciclo de manipulación es también un problema.

atracción magnética tiende a penetrar más allá de la lámina superior en la pila, lo que da lugar a la posibilidad de que más de una lámina sea elevada por el imán, problema que es posible superar con ciertas técnicas.

Las pinzas magnéticas pueden dividirse en dos categorías: las que utilizan electroimanes y las que emplean imanes permanentes. Las pinzas electromagnéticas son fáciles de controlar, pero requieren una fuente de alimentación de corriente continua y una unidad de control adecuada.

Los imanes permanentes tienen la ventaja de no requerir una fuente de alimentación exterior para accionar el imán. Sin embargo, existe una pérdida de control que acompaña a esta ventaja aparente, la cual se presenta en el momento en el que ha de liberarse la pieza, pues debe proporcionarse algún medio para separar la pieza del imán. El dispositivo utilizado se denomina eyector o despegador, teniendo la función de separar mecánicamente la pieza del imán.

Pinzas adhesivas

Diseños de pinzas en los que una sustancia adhesiva realiza la acción de agarre pueden utilizarse para manipular tejidos y otros materiales livianos. Los requisitos para los elementos a manipular son que deben agarrarse por un lado solamente y que no son adecuadas otras formas de agarre. Una de las limitaciones potenciales de una pinza adhesiva es que la sustancia adhesiva pierde su pegajosidad con un empleo repetido. En consecuencia, su fiabilidad como dispositivo de agarre disminuye con cada ciclo de operación sucesivo. Para superar esta limitación el material adhesivo se carga en la forma de una cinta continua en un mecanismo de alimentación que está unido a la muñeca del manipulador. El mecanismo de alimentación funciona de manera similar a un mecanismo de la cinta de una máquina de escribir.

Ganchos, cucharas y otros dispositivos diversos

Una diversidad de otros dispositivos puede utilizarse para agarrar piezas o materiales en las aplicaciones robóticas. Los ganchos pueden emplearse como efectores finales para manipular contenedores de piezas y para cargar y descargar piezas que cuelguen de transportadores aéreos.

Las cucharas y los calderos pueden utilizarse para manipular algunos materiales en forma de polvo o líquidos. Una de sus limitaciones es que la cantidad de material recogido por el manipulador es, a veces, difícil de controlar. El derrame durante el ciclo de manipulación es también un problema.

Otros tipos de pinzas incluyen dispositivos hinchables, en los que la vejiga o diafragma hinchable se expande para agarrar el objeto. La vejiga es de caucho u otro material elástico, lo que le hace para agarrar objetos frágiles. La pinza aplica una presión de agarre uniforme contra la superficie del objeto en lugar de una fuerza concentrada típica de una pinza mecánica.

Herramientas como efectores finales

En muchas aplicaciones se exige al manipulador que accione una herramienta en vez de una pieza de trabajo. En un número limitado de estas aplicaciones el efector final es una pinza que está diseñada para agarrar y manipular la herramienta. El motivo para utilizar una pinza en estas aplicaciones es que puede existir más de una herramienta a utilizar por el manipulador en el ciclo de trabajo. El empleo de una pinza permite que las herramientas se intercambien durante el ciclo y así se facilita esta función de manipulación multiherramienta.

En la mayoría de las aplicaciones de robot en las que se manipula una herramienta esta última está unida directamente a la muñeca del manipulador, en estos casos la herramienta es el efector final. Algunos ejemplos de herramientas utilizadas como efectores finales en aplicaciones de robot incluyen:

Herramientas de soldadura por puntos.
Soplete de soldadura por arco.
Tobera de pintura por pulverización.
Husillos giratorios para operaciones tales como:
- taladrado
- ranurado
- cepillado
- rectificado

Aplicadores de cemento líquido para montaje.
Sopletes de calentamiento.
Herramienta de corte por chorro de agua.

En cada caso el robot debe controlar la actuación de la herramienta.

2.5 CASO DE APLICACIÓN.

El Brazo de Electrónica Veneta.

El Mini-Brazo de Electrónica Veneta que fue utilizado para hacer pruebas, es un ejemplo de un brazo articulado industrial en escala reducida, con el articulado típico de los brazos industriales, el cual fue desarrollado para fines didácticos y experimentales, de modo que pudo ser conectado al sistema electrónico desarrollado, sirviendo como un periférico del sistema.

La geometría del Mini-Brazo ha sido proyectada para ofrecer la máxima flexibilidad junto con la máxima practicidad. El movimiento es efectuado en el momento en que es energizado alguno de los motores de pasos con los que cuenta. El brazo cuenta con seis articulaciones (6 motores de pasos), presentando seis grados de libertad.

El brazo se compone de 5 partes fundamentales.

- LA BASE, tiene la función de sostener la parte sobrante del brazo y contiene el motor que controla la rotación.

- EL HOMBRO, que rueda sobre la base por medio del engranaje principal, lleva cinco motores y sus reductores que se acoplan con los engranajes del antebrazo.

- EL BRAZO, que rueda alrededor de un eje horizontal del hombro. En la parte inferior se hallan los engranajes y los cables que ponen en movimiento el codo, la muñeca y la mano.

- EL ANTEBRAZO rueda alrededor de un eje horizontal del brazo y lleva los engranajes cónicos de la muñeca.

- MUÑECA y MANO. Los dos movimientos de la muñeca, la rotación alrededor del eje de la mano (twist) y la rotación de la mano alrededor del eje horizontal (up and down), dependen de la combinación de dos movimientos independientes. El movimiento de (twist) se obtiene moviendo los dos engranajes cónicos en dirección contraria, mientras que el movimiento de "arriba-abajo" se obtiene moviendo el par cónico en la misma dirección. Mientras que la mano, dotada de tres dedos con las extremidades de goma, puede abrirse o cerrarse.

Mando de las Fases de los Motores.

Los motores serán controlados directamente por las señales presentes en el puerto paralelo (82C55); esto permite el movimiento de paso completo o de medio paso. En las tablas siguientes pueden verse las secuencias que hay que enviar para los dos tipos de movimiento.

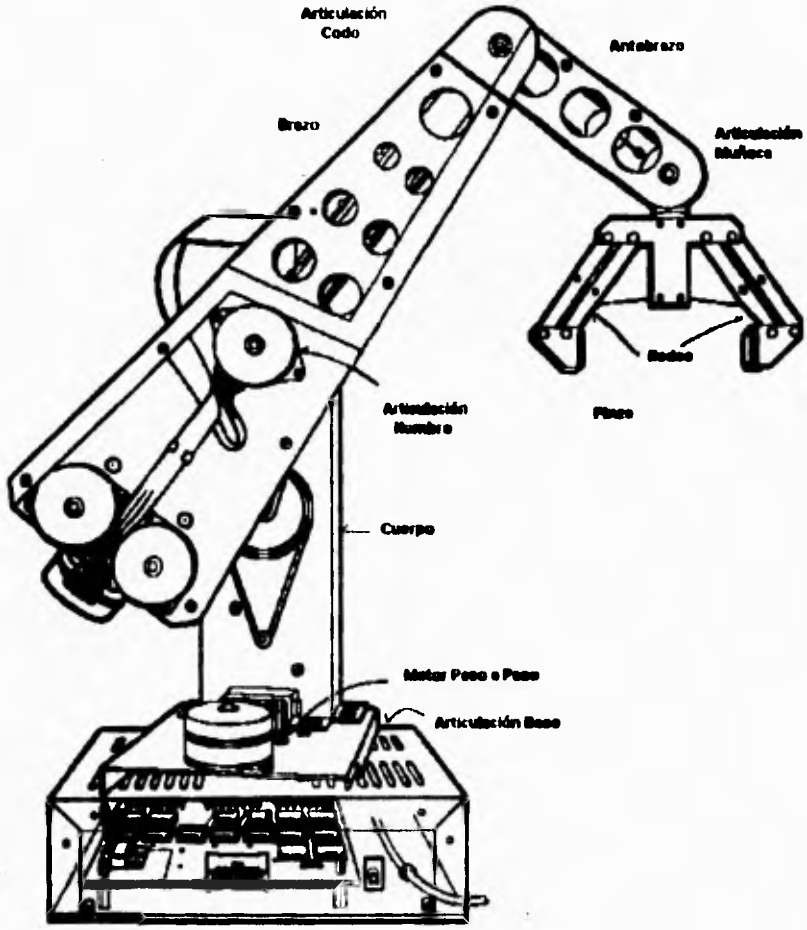


Figura 2.4. Manipulador mecánico en configuración de brazo articulado similar al Mini-Brazo de Electrónica **Veneta**.

MOVIMIENTO DE PASO COMPLETO

	MSB			LSB		
FASES (Bits):	A	B	C	D	PASO	DATO
	1	1	0	0	1	Ch
	1	0	0	1	2	9h
	0	0	1	1	3	3h
	0	1	1	0	4	6h

MOVIMIENTO DE MEDIO PASO

	MSB			LSB		
FASES (Bits):	A	B	C	D	PASO	DATO
	1	1	0	0	1	Ch
	1	0	0	0	1.5	8h
	1	0	0	1	2	9h
	0	0	0	1	2.5	1h
	0	0	1	1	3	3h
	0	0	1	0	3.5	2h
	0	1	1	0	4	6h
	0	1	0	0	4.5	4h

Relación entre el Paso del Motor y el Incremento Angular.

A continuación pueden verse los cálculos que permiten conocer la relación entre cada paso del motor y el incremento angular de cada articulación.

Estas constantes son necesarias si se desea construir un sistema de referencia cartesiano para los movimientos, o bien un sistema de referencia relacionado con las articulaciones del brazo.

BASE, MUÑECA y MANO

ángulo del paso x relación 1 x relación 2

$$7.5 \times \frac{20 \text{ dientes}}{72 \text{ dientes}} \times \frac{12 \text{ dientes}}{108 \text{ dientes}}$$

= 0.2314 grados por paso
= 4.32152 pasos por grado

HOMBRO y CODO

$$7.5 \times \frac{14 \text{ dientes}}{72 \text{ dientes}} \times \frac{12 \text{ dientes}}{108 \text{ dientes}}$$

= 0.162 grados por paso
= 6.17284 pasos por grado

MANO

$$7.5 \times \frac{20 \text{ dientes}}{72 \text{ dientes}} \times \frac{12 \text{ dientes}}{108 \text{ dientes}} = 0.231 \text{ grados por paso}$$

$$\frac{d \times 0.231}{360} = \frac{0.0524}{2} \text{ mm}$$

= 0.0262 mm de movimiento de la mano por paso del motor = 3.1415926

d = 26 mm = diámetro de la polea
Movimiento total de la mano medido a partir de la mano abierta hasta donde se cierra la mano: 20.0 mm
Angulo realizado por cada dedo = 50

$$\frac{50}{20.0 \text{ mm}} \times 0.0262 \text{ mm} = 0.0655 \text{ grados por paso o bien } 15.2672 \text{ pasos por grado}$$

2.6 DESCRIPCIÓN DEL MICROCONTROLADOR 8751H.

El circuito 8751H de Intel es un microcontrolador de 8 bits, el cual está basado en la arquitectura de la familia MCS-51 y es construido con tecnología HMOS.

Características Principales:

- Unidad Central de Proceso (CPU) de 8-bits
- Oscilador y un circuito de reloj internos
- 32 líneas de entrada/salida
- 64K de espacio de direcciones para memoria de datos externos
- 64K de espacio de direcciones para memoria de programación externa
- Dos Timer/Contadores de 16 bits
- Cinco fuentes de interrupción con dos niveles de prioridad
- Puerto serial full duplex
- Unida Lógica Aritmética (ALU)

Organización de Memoria.

El 8751 tiene separados los espacios de dirección para memoria de programación y memoria de datos. La memoria de programación está por arriba de los 64K bytes. Los 4K más bajos se encuentran internamente en el circuito. La memoria de datos puede consistir en 64K bytes de memoria externa, además de incluir 128 bytes de RAM internos, más los "SFRs" (Registros de Funciones Especiales) como se lista a continuación:

SÍMBOLO	NOMBRE	DIRECCIÓN
*ACC	Acumulador	0E0H
*B	Registro B	0F0H
*PSW	Palabra de estado de programación	0D0H
SP	Apuntador de pila	81H
DPTR	Apuntador de datos (consiste de DPH y DPL)	83H y 82H
*P0	Puerto 0	80H
*P1	Puerto 1	90H
*P2	Puerto 2	0A0H
*P3	Puerto 3	0B0H
*IP	Registro de control para la prioridad de las interrupciones	0B8H
*IE	Registro de control para habilitación de interrupciones	0A8H
TMOD	Modo de control del Timer/Contador	89H
TCON	Control del Timer/Contador	88H

TH0	Timer/Contador 0 (byte más significativo)	8CH
TL0	Timer/Contador 0 (byte menos significativo)	8AH
TH1	Timer/Contador 1 (byte más significativo)	8DH
TL1	Timer/Contador 1 (byte menos significativo)	8BH
*SCON	Control serial	98H
SBUF	Almacenador serial de datos	99H
PCON	Control de encendido	97H

En los registros de funciones especiales marcados con un asterisco pueden direccionarse los 8 bits al mismo tiempo o direccionar un solo bit, según se desee.

Descripción de las terminales del 8751H.

Vss. Potencial negativo del circuito.

Vcc. Voltaje proporcionado durante la programación, verificación y operación normal.

Puerto 0. El puerto 0 es un puerto bidireccional I/O de 8 bit con colector abierto. También multiplexa las direcciones bajas y los datos del bus durante los accesos a memoria externa. También recibe las instrucciones durante la programación de la EPROM, y los bytes de instrucción de salida durante la verificación del programa. El puerto 0 puede absorber (y en operación como bus puede entregar) 8 entradas LS TTL.

Puerto 1. El puerto 1 es un puerto bidireccional I/O de 8 bits con resistencias internas de colector abierto. Recibe los bytes de direcciones de bajo órden durante la programación de la EPROM y la verificación de programa. El puerto puede absorber/entregar cuatro entradas LS TTL.

Puerto 2. Es un puerto bidireccional I/O de 8 bits con resistencias internas de colector abierto. Emite los bytes de direcciones de alto órden durante el acceso a memoria externa. También recibe los bits de direcciones de alto órden durante la programación de la EPROM y verificación de programa. El puerto 2 puede absorber/entregar cuatro entradas LS TTL.

Puerto 3. Es un puerto bidireccional I/O de 8 bits con resistencias internas de colector abierto. También proporciona las funciones de varias características especiales de la familia del MCS-51, como se lista a continuación:

Terminales del Puerto	Función Alterna
P3.0	RXD (Entrada de puerto serial)
P3.1	TXD (Salida de puerto serial)
P3.2	INT0 (Interrupción externa)
P3.3	INT1 (Interrupción externa)

P3.4	T0 (Timer/Contador 0, entrada externa)
P3.5	T1 (Timer/Contador 1, entrada externa)
P3.6	WR (Señal de habilitación de escritura para la memoria externa de datos)
P3.7	RD (Señal de habilitación de lectura para la memoria externa datos)

Las funciones alternas solamente pueden ser activadas si el bit correspondiente es mantenido en el registro especial del puerto conteniendo un uno. De otra manera la terminal del puerto tendrá un cero. El puerto 3 puede absorber/entregar cuatro entradas LS TTL.

RST. Un uno en esta terminal durante dos ciclos de máquina mientras el oscilador está corriendo inicializa el dispositivo.

ALE/PROG(Negado). Habilitador del "latch" de direcciones de salida para guardar el byte más bajo de las direcciones durante el acceso de memoria externa. ALE es activado con una velocidad constante de 1/6 de la frecuencia de oscilación, excepto durante un acceso de datos de memoria externo en el cual la primera vez el pulso de ALE se omite. ALE puede absorber/entregar 8 entradas LS TTL. Esta terminal también es el pulso de entrada de programa (PROG negado) durante la programación de la EPROM.

PSEN(negado). El habilitador de salida de estado de programa (PSEN) es el que activa la lectura del programa en una memoria externa. PSEN es activado dos veces cada ciclo de máquina durante intervalos de búsqueda para programas en memoria externa. (Durante la ejecución externa de programa dos activaciones de PSEN son omitidas durante cada acceso a datos de memoria externos). PSEN no es activado durante intervalos de búsqueda para programa en memoria interna. PSEN puede absorber/entregar 8 entradas LS TTL.

EA(negada)/Vpp. Cuando EA es colocada en alto, el 8751H ejecuta del programa de la memoria interna (a menos que el contador de programa exceda la dirección 0FFFH), Cuando EA esta en bajo, el 8751 ejecuta solamente el programa de la memoria externa. Esta terminal también recibe los 21 volts de programación (VPP) durante la programación de la EPROM. Esta terminal no debe estar flotando durante la operación normal.

XTAL1. Entrada al amplificador invertido que forma el oscilador. XTAL1 debe ser tierra cuando es usado un oscilador externo.

XTAL2. Salida del amplificador invertido que forma el oscilador, y entrada al generador de reloj interno. Recibe las señales de oscilación externas cuando un oscilador externo es usado.

Oscilador y Circuito de Reloj.

XTAL1 y XTAL2 son las entradas y las salidas de una etapa inversora simple en el microcontrolador, el cual puede ser configurado con componentes externos como un oscilador "Pierce" regulado por cristal (ver circuito de reloj en el diagrama eléctrico). El oscilador maneja el generador interno de reloj, el cual provee las señales de reloj internas al integrado. Las señales internas de reloj son un medio de la frecuencia de oscilación, y definen las fases internas, estados, y ciclos de máquina.

Tiempos en el CPU.

Un ciclo de máquina consta de seis estados (12 períodos de oscilación). Cada estado es dividido en dos, una mitad corresponde a la Fase 1, durante la cual el reloj de la fase 1 es activo, y la otra mitad corresponde a la Fase 2, durante la cual el reloj de la fase 2 es activo. Así, un ciclo de máquina consiste de 12 períodos de oscilación, numerados S1P1 (Estado 1, Fase 1), hasta S6P2 (Estado 6, Fase 2). Cada fase dura un período de oscilación. Cada estado dura dos períodos de oscilación. Comúnmente, operaciones aritméticas y lógicas se llevan a cabo durante la Fase 1 y transferencias internas de registro a registro se llevan a cabo durante la Fase 2.

Característica al Leer-Modificar-Escribir.

Algunas instrucciones que leen un puerto leen el "latch" y otras leen la terminal ("pin"). Las instrucciones que leen el "latch" en vez de la terminal son las que leen un valor, o la posibilidad de que este cambie, y lo reescriben en el "latch". Estas son llamadas instrucciones de "leer-modificar-escribir". Las instrucciones listadas a continuación son instrucciones de este tipo. Cuando el destino es un puerto, o un bit del puerto, estas instrucciones leen el "latch" en vez de la terminal:

ANL	(AND lógica, ejem., ANL P1,A)
ORL	(OR lógica, ejem., ORL P2,0)
XRL	(EX-OR lógica, ejem., XRL P3,A)
JBC	(Salta si bit=1 y limpia bit, ejem., JBC P1.1, Etiqueta)
CPL	(Complementa bit, ejem., CPL P3,0)
INC	(Incrementa, ejem., INC P2)
DEC	(Decrementa, ejem., DEC P2)
DJNZ	(Decrementa y salta si no es cero, ejem., DJNZ P3, Etiqueta)
MOV PX.Y,C	(Mueve el bit de acarreo al bit Y del puerto X)
CLR PX.Y	(Limpia el bit Y del puerto X)
SET PX.Y	(Habilita el bit Y del puerto X)

NO es obvio que las tres últimas instrucciones de esta lista son instrucciones de leer-modificar-escribir, pero lo son. Estas leen el byte del puerto, los 8 bits, modificando el bit direccionado, y escribiendo de nuevo el byte en el flip flop de mantenimiento ("latch").

La razón de que este tipo de instrucciones son dirigidas al "latch" en lugar de la terminal es para evitar una posible interpretación errónea del nivel de voltaje de la terminal. Por ejemplo, un bit del puerto puede ser usado para manejar la base del transistor. Cuando un uno es escrito en el bit, el transistor es encendido. Si el CPU lee el mismo bit del puerto en la terminal, en lugar del "latch", leerá el voltaje de la base del transistor y lo interpretará como un cero. Leyendo el "latch" en lugar de la terminal regresará el valor correcto de 1.

Accesando Memoria Externa.

Los accesos a memoria externa son de dos tipos: accesos a memoria de programa externa y acceso a memoria de datos externa. Accesos a memoria de programa externa usa la señal PSEN (negada) (Habilitador de programa) como el habilitador de lectura. Accesos de memoria de datos externa usa RD (negada) o WR (negada) (funciones alternas de P3.7 y P3.6) para habilitar la memoria.

Accesos desde la memoria de programa externa siempre usan una dirección de 16 bits. Accesos a la memoria de datos externa pueden usar una dirección de 16 bits (MOVX @DPTR) o una dirección de 8 bits (MOVX @Ri).

Siempre que una dirección de 16 bits es usada, el byte más alto de la dirección sale al Puerto 2, donde es mantenido por la duración del ciclo de lectura o escritura. Durante este tiempo el "latch" del Puerto 2 (el Registro de Función Especial) no tiene que contener unos, y el contenido del SFR del Puerto 2 no es modificado. Si el ciclo de la memoria externa no es inmediatamente seguido por otro ciclo de la memoria externa, el contenido no perturbado del SFR del Puerto 2 aparecerá de nuevo en el siguiente ciclo.

Si una dirección de 8 bits es usada (MOVX @Ri), el contenido del SFR del Puerto 2 permanecerá en las terminales del Puerto 2 durante el ciclo de memoria externo, lo que facilitara la paginación.

En cualquier caso, el byte más bajo de la dirección es multiplexado en tiempo con el byte de datos del Puerto 0. La señal de DIRECCIÓN/DATOS maneja los FETs (transistores de efecto de campo) en el bufer de salida del Puerto 0. Así, en estas aplicaciones las terminales del Puerto 0 no son salidas en colector abierto por lo que no se requieren resistencias externas en esta

NO es obvio que las tres últimas instrucciones de esta lista son instrucciones de leer-modificar-escribir, pero lo son. Estas leen el byte del puerto, los 8 bits, modificando el bit direccionado, y escribiendo de nuevo el byte en el flip flop de mantenimiento ("latch").

La razón de que este tipo de instrucciones son dirigidas al "latch" en lugar de la terminal es para evitar una posible interpretación errónea del nivel de voltaje de la terminal. Por ejemplo, un bit del puerto puede ser usado para manejar la base del transistor. Cuando un uno es escrito en el bit, el transistor es encendido. Si el CPU lee el mismo bit del puerto en la terminal, en lugar del "latch", leerá el voltaje de la base del transistor y lo interpretará como un cero. Leyendo el "latch" en lugar de la terminal regresará el valor correcto de 1.

Accesando Memoria Externa.

Los accesos a memoria externa son de dos tipos: accesos a memoria de programa externa y acceso a memoria de datos externa. Accesos a memoria de programa externa usa la señal PSEN (negada) (Habilitador de programa) como el habilitador de lectura. Accesos de memoria de datos externa usa RD (negada) o WR (negada) (funciones alternas de P3.7 y P3.6) para habilitar la memoria.

Accesos desde la memoria de programa externa siempre usan una dirección de 16 bits. Accesos a la memoria de datos externa pueden usar una dirección de 16 bits (MOVX @DPTR) o una dirección de 8 bits (MOVX @Ri).

Siempre que una dirección de 16 bits es usada, el byte más alto de la dirección sale al Puerto 2, donde es mantenido por la duración del ciclo de lectura o escritura. Durante este tiempo el "latch" del Puerto 2 (el Registro de Función Especial) no tiene que contener unos, y el contenido del SFR del Puerto 2 no es modificado. Si el ciclo de la memoria externa no es inmediatamente seguido por otro ciclo de la memoria externa, el contenido no perturbado del SFR del Puerto 2 aparecerá de nuevo en el siguiente ciclo.

Si una dirección de 8 bits es usada (MOVX @Ri), el contenido del SFR del Puerto 2 permanecerá en las terminales del Puerto 2 durante el ciclo de memoria externo, lo que facilitara la paginación.

En cualquier caso, el byte más bajo de la dirección es multiplexado en tiempo con el byte de datos del Puerto 0. La señal de DIRECCIÓN/DATOS maneja los FETs (transistores de efecto de campo) en el bufer de salida del Puerto 0. Así, en estas aplicaciones las terminales del Puerto 0 no son salidas en colector abierto por lo que no se requieren resistencias externas en esta

función. La señal ALE (Habilitador del "latch" de direcciones) debe ser usado para capturar el byte de direcciones dentro del "latch" externo. El byte de direcciones es válido en la transición negativa de ALE. Entonces, en un ciclo de escritura, el byte de datos escrito aparece en el Puerto 0 antes de que se active la señal de escritura (WR negada), y permanece ahí hasta después de que WR (negada) es desactivada. En un ciclo de lectura, el byte entrante es aceptado en el Puerto 0 antes de que el habilitador de lectura es desactivado.

Durante cualquier acceso a memoria externa, el CPU escribe 0FFH en el "latch" del Puerto 0 (en el Registro de Función Especial), con lo que es borrada cualquier información que pudiera haber retenido el Registro de Función Especial del Puerto 0.

Memoria de Programa Externa es accesada bajo dos condiciones:

1) Durante cualquier activación de la señal EA (Negada); o
2) Cualquier contenido del contador de programa (PC) superior a 0FFFH.

Esto requiere que las versiones de ROM tengan el EA (negado) alambrado en bajo para habilitar los 4K bytes más bajos de programa para ser accesados desde memoria externa.

Cuando el CPU está ejecutando una Memoria de Programa externa, los 8 bits del Puerto 2 son dedicados como una función de salida y no pueden ser usados para propósito general de I/O. Durante accesos al programa externo, estos envían el byte alto del PC, y durante accesos a Memoria de Datos externa, estos mandan DPH o el SFR del Puerto 2 (dependiendo de si el acceso a Memoria de Datos externa es un MOVX @DPTR o un MOVX @Ri).

Timer/Contadores.

El 8751 tiene dos registros de 16 bits que pueden ser configurados para operar como timers o también como contadores: Timer 0 y Timer 1.

En la función de "timer", el registro es incrementado cada ciclo de máquina. Así uno puede imaginárselo como un contador de ciclos de máquina. Debido a que un ciclo de máquina consiste de 12 períodos de oscilación, la velocidad de conteo es de 1/12 de la frecuencia de oscilación.

En la función de "contador", el registro es incrementado en el momento en que se presenta una transición de 1 a 0 en la entrada externa de la correspondiente terminal, T0 o T1. En esta función, la entrada externa es mostrada durante S5P2 o cada ciclo de máquina. Cuando aparece un alto en un ciclo, y en el siguiente ciclo un bajo, es el momento en el que es incrementada la cuenta. El nuevo valor de la cuenta se presenta en el registro durante S3P1, que es el ciclo posterior a la aparición de un uno, momento en que la transición es detectada. A partir del momento en que ocurren dos ciclos de máquina (24 períodos de oscilación) es reconocida la transición de 1 a 0; la máxima velocidad de conteo es 1/24 de la frecuencia de oscilación. Para asegurar que un nivel dado es mostrado al menos una vez antes de que cambie, este debe ser retenido por lo menos un ciclo completo de máquina.

Adicionalmente a la selección como "Timer" o "Contador", El Timer 0 y el Timer 1 tienen cuatro modos de operación a elegir. La función de timer o contador se selecciona con los bits de control C/T en el registro de función especial TMOD. Los cuatro modos de operación son elegidos por la pareja de bits (M1, M0) que se encuentran en TMOD.

(MSB)							(LSB)
GATE	C/T	M1	M0	GATE	C/T	M1	M0
		Timer 1			Timer 0		

TMOD: Registro de modo de control del Timer/Contador.

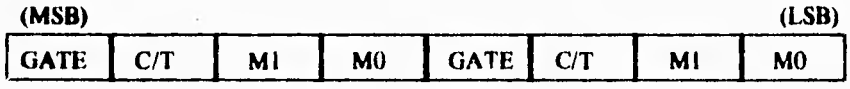
GATE Control de conmutación cuando es habilitado. El Timer/Contador "x" es habilitado solamente si la terminal INTx es alto y la terminal de control TRx es activado. Cuando es bajo el Timer "x" es activado si TRx es alto.

C/T Selector de operación como Timer o Contador, cuando está desactivado opera como Timer (la entrada es por medio del sistema de reloj interno). Cuando es habilitado trabaja como Contador (la entrada es por medio de la terminal Tx).

M1	M0	Modos de Operación
0	0	Timer MCS-48, El registro TLx sirve como un pre-escalar de cinco bits.
0	1	Timer/Contador de 16 bits, THx y TLx están en cascada; no existe un pre-escalar.
1	0	Timer-contador de 8 bits recargable automáticamente. THx retiene el valor que será recargado en TLx cada vez que llegue a su tope el timer-contador.

En la función de "contador", el registro es incrementado en el momento en que se presenta una transición de 1 a 0 en la entrada externa de la correspondiente terminal, T0 o T1. En esta función, la entrada externa es mostrada durante S5P2 o cada ciclo de máquina. Cuando aparece un alto en un ciclo, y en el siguiente ciclo un bajo, es el momento en el que es incrementada la cuenta. El nuevo valor de la cuenta se presenta en el registro durante S3P1, que es el ciclo posterior a la aparición de un uno, momento en que la transición es detectada. A partir del momento en que ocurren dos ciclos de máquina (24 períodos de oscilación) es reconocida la transición de 1 a 0; la máxima velocidad de conteo es 1/24 de la frecuencia de oscilación. Para asegurar que un nivel dado es mostrado al menos una vez antes de que cambie, este debe ser retenido por lo menos un ciclo completo de máquina.

Adicionalmente a la selección como "Timer" o "Contador", El Timer 0 y el Timer 1 tienen cuatro modos de operación a elegir. La función de timer o contador se selecciona con los bits de control C/T en el registro de función especial TMOD. Los cuatro modos de operación son elegidos por la pareja de bits (M1, M0) que se encuentran en TMOD.



TMOD: Registro de modo de control del Timer/Contador.

GATE Control de conmutación cuando es habilitado. El Timer/Contador "x" es habilitado solamente si la terminal INTx es alto y la terminal de control TRx es activado. Cuando es bajo el Timer "x" es activado si TRx es alto.

C/T Selector de operación como Timer o Contador, cuando está desactivado opera como Timer (la entrada es por medio del sistema de reloj interno). Cuando es habilitado trabaja como Contador (la entrada es por medio de la terminal Tx).

M1	M0	Modos de Operación
0	0	Timer MCS-48, El registro TLx sirve como un pre-escalar de cinco bits.
0	1	Timer/Contador de 16 bits, THx y TLx están en cascada; no existe un pre-escalar.
1	0	Timer-contador de 8 bits recargable automáticamente. THx retiene el valor que será recargado en TLx cada vez que llegue a su tope el timer-contador.

1 1	(Timer 0) TL0 es un timer-contador de 8 bits, controlado por regla por los bits de control del Timer 0. TH0 es un timer de 8 bits, el cual solo puede ser controlado con los bits de control del Timer 1.
1 1	(Timer 1) Timer-contador 1 detenido.

Los Modos 0,1 y 2 funcionan igual para el timer o el contador, pero en el Modo 3 se tienen algunas diferencias.

Descripción de los Modos de Operación.

MODO 0

Seleccionando el Timer en el modo 0 se logra que trabaje como un Timer 8048, el cual es un contador de 8 bits dividido por un pre-escalar de 32.

En este modo, el registro del timer es configurado como un registro de 13 bits. Como el conteo desplaza todos los unos por ceros, esto funciona como la bandera de interrupción TF1 del timer. La entrada de conteo es habilitada en el Timer cuando TR1 = 1 y el GATE (compuerta) = 0 o INT1 = 1. (Colocar GATE = 1 permite que el Timer sea controlado por la entrada externa INT1, para facilitar las mediciones del ancho del pulso). TR1 es un bit de control en el registro de función especial TCON. El GATE esta presente en TMOD.

El registro de trece bits consiste de los 8 bits de TH1 y de los 5 bits más bajos de TL1. Los 3 bits altos de TL1 son indeterminados y deben ser ignorados. Al seleccionar la bandera de ejecución (TR1) los registros no son limpiados.

(MSB) (LSB)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TCON: Registro de control del Timer/Contador.

Símbolo	Posición	Nombre y Significado
TF1	TCON.7	Bandera del Timer 1 para indicar un sobre conteo. Habilitada por Hardware en el momento en el que el timer/contador presenta un sobre conteo. Deshabilitada por Hardware cuando es accesada la rutina de interrupción.

TR1	TCON.6	Bits de control para ejecución del Timer 1. Habilitado/deshabilitado por software para encender o apagar el timer/contador.
TF0	TCON.5	Bandera del Timer 0 para indicar un sobre conteo. Habilitada por hardware en el momento en el que el timer/contador presenta un sobre conteo. Deshabilitada por Hardware cuando es accesada la rutina de interrupción
TR0	TCON.4	Bits de control para ejecución del Timer 0. Habilitado/deshabilitado por software para encender o apagar el timer/contador.
IE1	TCON.3	Bandera de flanco de la Interrupción 1. Habilitada por hardware cuando el flanco de bajada de la interrupción externa es detectado. Es deshabilitada cuando la interrupción es procesada.
IT1	TCON.2	Bit para especificar el tipo de control de la Interrupción 1. Manejada por software para especificar activación con flanco de bajada/con nivel bajo de la interrupción externa 1.
IE0	TCON.1	Bandera de flanco de la Interrupción 0. Habilitada por hardware cuando el flanco de bajada de la interrupción externa es detectado. Es deshabilitada cuando la interrupción es procesada.
IT0	TCON.0	Bit para especificar el tipo de control de la Interrupción 0. Manejada por software para especificar activación con flanco de bajada/con nivel bajo de la interrupción externa 0.

La operación en Modo 0 es la misma para el Timer 0 como para el Timer 1. Existen dos diferentes bits GATE, uno para el Timer 1 (TMOD.7) y otro para el Timer 0 (TMOD.3).

MODO 1

El Modo 1 es igual al Modo 0, excepto porque el registro del Timer usa en su ejecución los 16 bits.

MODO 2

El Modo 2 configura el registro del timer como un contador de 8 bits (TL1) con recarga automática. En el momento en que llega a su máximo valor el registro TL1, no solamente se

habilita la bandera TF1, sino que también se recarga TL1 con el contenido de TH1, el cual es preestablecido por software. La recarga deja TH1 sin ningún cambio.
La operación en Modo 2 es la misma para el Timer/Contador 0.

Modo 3.

Timer 1 en Modo 3 simplemente mantiene su cuenta. El efecto es el mismo si a TR1 le asignamos el valor de cero.
El Timer 0 en Modo 3 establece TL0 y TH0 como dos contadores separados.
El Modo 3 es provisto para aplicaciones que requieren un timer o contador extra de 8 bits. Con el Timer 0 en Modo 3, un 8751 puede trabajar como si tuviera tres timers/contadores. Cuando el Timer 0 está en Modo 3, el Timer 1 puede estar encendido y apagado conmutándolo fuera y dentro de su propio Modo 3, o puede seguirse usando con el puerto serial para generar la velocidad de transmisión, o en cualquier aplicación que no sea requerida una interrupción.

Interfase Serial

El puerto serial es full duplex, lo que significa que la transmisión y la recepción pueden ser simultáneas. También trabaja como un almacenador en recepción, lo que quiere decir que puede comenzar a recibir un segundo byte antes de que la recepción de un byte previo halla sido leída del registro receptor. (No obstante, si el primer byte continúa sin ser leído y es completado el tiempo de recepción del segundo byte, uno de los bytes se perderá).

Los registros del puerto serial para transmisión y recepción son ambos accesados por el registro de función especial SBUF. Al escribir en el SBUF se carga el registro de transmisión y cuando se lee SBUF se accesa un registro físico de recepción separado.

El puerto serial puede operar en cuatro modos:

Modo 0: La entrada y salida del dato serial es a través de RXD. TXD muestra la variación del timer. 8 bits son transmitidos/recibidos: 8 bits de datos (LSB primero). La velocidad de transmisión es fijada a un doceavo de la frecuencia de oscilación.

Modo 1: 10 bits son transmitidos (a través de TXD) o recibidos (a través de RXD): un bit de inicio (0), 8 bits de datos (LSB primero), y un bit de paro (1). En recepción, el bit de paro es localizado en RB8 en el registro de función especial SCON. La velocidad de transmisión es variable, la cual es determinada en el 8751H por el Timer 1 a velocidad de saturación.

Modo 2: 11 bits son transmitidos (a través de TXD) o recibidos (a través de RXD): un bit de inicio (0), 8 bits de datos (LSB primero), un noveno bit de dato programable y un bit de paro (1). En transmisión, al noveno bit de dato (TB8 en SCON) se le pude asignar el valor de 0 o 1. O por ejemplo, el bit de paridad (P, en el PSW) puede ser movido dentro de TB8. En recepción el noveno bit de dato es localizado en RB8 en el registro de función especial SCON, mientras que el bit de paro es ignorado. La velocidad de transmisión es programada a 1/32 o 1/64 de la frecuencia de oscilación.

Modo 3: 11 bits son transmitidos (a través de TXD) o recibidos (a través de RXD): un bit de inicio (0), 8 bits de datos (LSB primero), un noveno bit de dato programable y un bit de paro (1). De hecho, el Modo 3 es lo mismo que el Modo 2 excepto por la velocidad de transmisión. La velocidad de transmisión en el Modo 3 es variable.

En los cuatro modos, la transmisión es iniciada por alguna instrucción que utilice SBUF como un registro de destino. Recepción es iniciada en el Modo 0 por la condición RI=0 y REN=1. La recepción es iniciada en los otros modos al recibirse el bit de inicio si REN=1.

Registro de Control del Puerto Serial.

El control del puerto serial y el estado de los registros se efectúa por medio del Registro de Función Especial SCON. Este registro contiene no solamente los bits de modos de selección, sino también el noveno bit de dato para transmisión y recepción (TB8 y RB8), y los bits de interrupción del puerto serial (TI y RI).

SCON: Registro de control del Puerto Serial.

donde SM0, SM1 especifican el modo del puerto serial, como se explicará a continuación.

SM0	SM1	MODO	Descripción	Velocidad De Transmisión
0	0	0		$f_{osc.}$ /12
0	1	1	8 bit UART	variable
1	0	2	9 bit UART	$f_{osc.}$ /64 o $f_{osc.}$ /12
1	1	3	9 bit UART	variable

- **SM2** Habilita las características de comunicación del multiprocesador en los modos 2 y 3. En el modo 2 o 3, si SM2 es puesto a 1 entonces RI no será activada si el noveno bit de dato recibido (RB8) es cero. En modo 1, si SM2 = 1 entonces RI no será activada si un bit de paro no es recibido. En modo 0, SM2 debe ser cero.

- **REN** Habilita la recepción serial. Se debe activar por software para habilitar la recepción. Desactivar por software para deshabilitar la recepción.

- **TB8** Es el noveno bit de dato que puede ser transmitido en los modos 2 y 3. Habilitado o deshabilitado por software cuando se desee.

- **RB8** En modos 2 y 3, es el noveno bit de datos que es recibido. En modo 1, si SM2 = 0, RB8 es el bit de paro que es recibido. En modo 0, RB8 no es usado.

- **TI** Es la bandera de interrupción transmitida. Habilitada por hardware al final del ciclo del octavo bit en el modo 0, o durante la mitad del ciclo del bit de paro en los otros modos, en cualquier transmisión serial (excepto en SM2). Debe ser desactivada por software.

- **RI** Es la bandera de interrupción recibida. Habilitada por hardware al final del ciclo del octavo bit en el modo 0, o durante la mitad del ciclo del bit de paro en los otros modos, en cualquier recepción serial (excepto en SM2). Debe ser desactivado por software.

Velocidad de Transmisión.

La velocidad de transmisión en el Modo 0 es fijada:

$$\text{Velocidad de Transmisión en Modo 0} = \frac{\text{Frecuencia de Oscilación}}{12}$$

La velocidad de transmisión en el Modo 2 depende del valor del bit SMOD en el Registro de Función Especial PCON. Si SMOD = 0 (que es el valor en el reset), la velocidad de transmisión es 1/64 de la frecuencia de oscilación. Si SMOD = 1, la velocidad de transmisión es 1/32 de la frecuencia de oscilación.

$$\text{Velocidad de Transmisión en Modo 2} = \frac{2^{SMOD}}{64} x(\textit{Frec. de Oscilación})$$

En el 8751, la velocidad de transmisión en el Modo 1 y 3 es determinada por el Timer 1 en velocidad de saturación.

Usando El Timer 1 para Generar Velocidades de Transmisión.

Cuando el Timer 1 es usado como el generador de la velocidad de transmisión, esta es determinada por el Timer 1 en velocidad de saturación y el valor de SMOD como se muestra:

Modos 1 y 3

$$\text{Vel. de Transmisión} = \frac{2^{SMOD}}{32} x(\textit{Reloj}\ \ 1\ \textit{Overflow Rate})$$

La interrupción del Timer 1 debe ser deshabilitada en esta aplicación. El Timer por sí mismo puede configurarse para operar como "Timer" o "Contador", y en cualquiera de sus tres modos de operación. En las aplicaciones más comunes, es configurado para operar como "Timer", en

Velocidad de Transmisión.

La velocidad de transmisión en el Modo 0 es fijada:

$$\text{Velocidad de Transmisión en Modo 0} = \frac{\text{Frecuencia de Oscilación}}{12}$$

La velocidad de transmisión en el Modo 2 depende del valor del bit SMOD en el Registro de Función Especial PCON. Si SMOD = 0 (que es el valor en el reset), la velocidad de transmisión es 1/64 de la frecuencia de oscilación. Si SMOD = 1, la velocidad de transmisión es 1/32 de la frecuencia de oscilación.

$$\text{Velocidad de Transmisión en Modo 2} = \frac{2^{SMOD}}{64} \times (\textit{Frec. de Oscilación})$$

En el 8751, la velocidad de transmisión en el Modo 1 y 3 es determinada por el Timer 1 en velocidad de saturación.

Usando El Timer 1 para Generar Velocidades de Transmisión.

Cuando el Timer 1 es usado como el generador de la velocidad de transmisión, esta es determinada por el Timer 1 en velocidad de saturación y el valor de SMOD como se muestra:

Modos 1 y 3

$$\text{Vel. de Transmisión} = \frac{2^{SMOD}}{32} \times (\textit{Reloj}\ \ 1\ \textit{Overflow Rate})$$

La interrupción del Timer 1 debe ser deshabilitada en esta aplicación. El Timer por sí mismo puede configurarse para operar como "Timer" o "Contador", y en cualquiera de sus tres modos de operación. En las aplicaciones más comunes, es configurado para operar como "Timer", en

el modo de auto recarga (la parte alta del registro TMOD = 0010B). En este caso, la velocidad de transmisión es dada por la fórmula

$$\text{Vel. de Transmisión en Modos 1, 3} = \frac{2^{SMOD}}{32} \times \frac{\textit{Frec. de Oscilación}}{12 \times [256 - (TH1)]}$$

Uno puede lograr velocidades de transmisión muy bajas con el Timer 1, dejando habilitada la interrupción del Timer 1, y configurándolo para trabajar como un timer de 16 bits (la parte alta del registro TMOD = 0001B), además de usar la interrupción del Timer 1 para realizar una sobrecarga de 16 bits por software. La Tabla 2.1 lista algunos de los valores más frecuentes de velocidades de transmisión que son usados y cómo pueden ser obtenidos a partir del Timer 1.

		Timer 1			
Vel. Transmisión	f_{osc}	SMOD	C/T	MODO	Valor de Recarga
Modo 0 Max: 1 MHz	12 MHz	X	X	X	X
Modo 2 Max: 375K	12 MHz	1	X	X	X
Modos 1, 3: 62.5K	12 MHz	1	0	2	FFH
19.2K	11.059 MHz	1	0	2	FDH
9.6K	11.059 MHz	0	0	2	FDH
4.8K	11.059 MHz	0	0	2	FAH
2.4K	11.059 MHz	0	0	2	F4H
1.2K	11.059 MHz	0	0	2	E8H
2.4K	6 MHz	1	0	2	F3H
1.2K	6 MHz	0	0	2	F3H
600	6 MHz	0	0	2	E6H
300	6 MHz	0	0	2	CCH
150	6 MHz	0	0	2	98H
110	6 MHz	0	0	2	72H
4.8K	12 MHz	1	0	2	F3H
2.4K	12 MHz	0	0	2	F3H
1.2K	12 MHz	0	0	2	E6H
600	12 MHz	0	0	2	CCH
300	12 MHz	0	0	2	98H
150	12 MHz	0	0	2	30H
110	12 MHz	0	0	1	FEEBH

Tabla 2.1.

Operación detallada del puerto serial en Modo 1.

La transmisión es iniciada por cualquier instrucción que use SBUF como un registro de destino. La señal de "escritura a SBUF" carga un 1 en el noveno bit del registro de transmisión y le indica a la unidad de Control TX que una transmisión es solicitada. La transmisión de hecho comienza en el ciclo de máquina S1P1 siguiendo el próximo desplazamiento en el contador el cual es dividido por 16. (Pero, los ciclos de los bit son sincronizados por el contador dividido por 16, no por la señal de "escritura a SBUF").

La Figura 2.5 muestra un diagrama de funcionamiento simplificado del puerto serial en Modo 1, y los tiempos asociados para la transmisión y recepción.

La transmisión empieza con la activación del SEND (negado), el cual coloca el bit de inicio en TXD. Para el siguiente ciclo, la señal DATA es activada, la cual habilita el bit de salida del registro de transmisión (registro de corrimiento) a TXD. El primer pulso de corrimiento ocurre un ciclo después de esto.

Como los bits de datos corren hacia fuera por la derecha, ceros son impulsados desde la izquierda. Cuando el bit más significativo del byte de datos está en la posición de salida del registro de corrimiento, entonces el 1 que fue inicialmente cargado en la novena posición está justo a la izquierda del bit más significativo, y todas las posiciones a la izquierda de este contienen ceros. Con esta bandera de condición, la unidad de Control TX realiza un último corrimiento y entonces desactiva SEND y activa TI. Esto ocurre en el décimo desplazamiento después de "escribir a SBUF".

La recepción es inicializada al detectarse una transición de cero a uno en RXD. Para este propósito RXD es muestreada a una velocidad de 16 veces la velocidad de transmisión (sin importar qué velocidad haya sido seleccionada). Cuando una transición es detectada, el contador (divido entre 16) es inmediatamente inicializado y un 1FFH es escrito dentro del registro de corrimiento de entrada. Inicializando el contador se ajusta su desplazamiento con los linderos del ciclo entrante.

Los 16 estados del contador dividen cada tiempo que dura el ciclo del bit en 16 partes. En el séptimo, octavo y noveno estados del contador de cada tiempo que dura el ciclo, el detector de bit comprueba el valor de RXD. El valor aceptado es el valor adquirido en al menos 2 de las tres muestras. Esto es realizado para tener inmunidad al ruido. Si el valor aceptado durante la presencia del primer bit no es cero, el circuito de recepción es inicializado y la unidad continúa buscando otra transición de 1 a 0. Esto se realiza con el objeto de tener inmunidad a falsos bits de inicio. Si el bit de inicio es válido, es desplazado dentro del registro de corrimiento de entrada, y la recepción del resto de la información se efectúa.

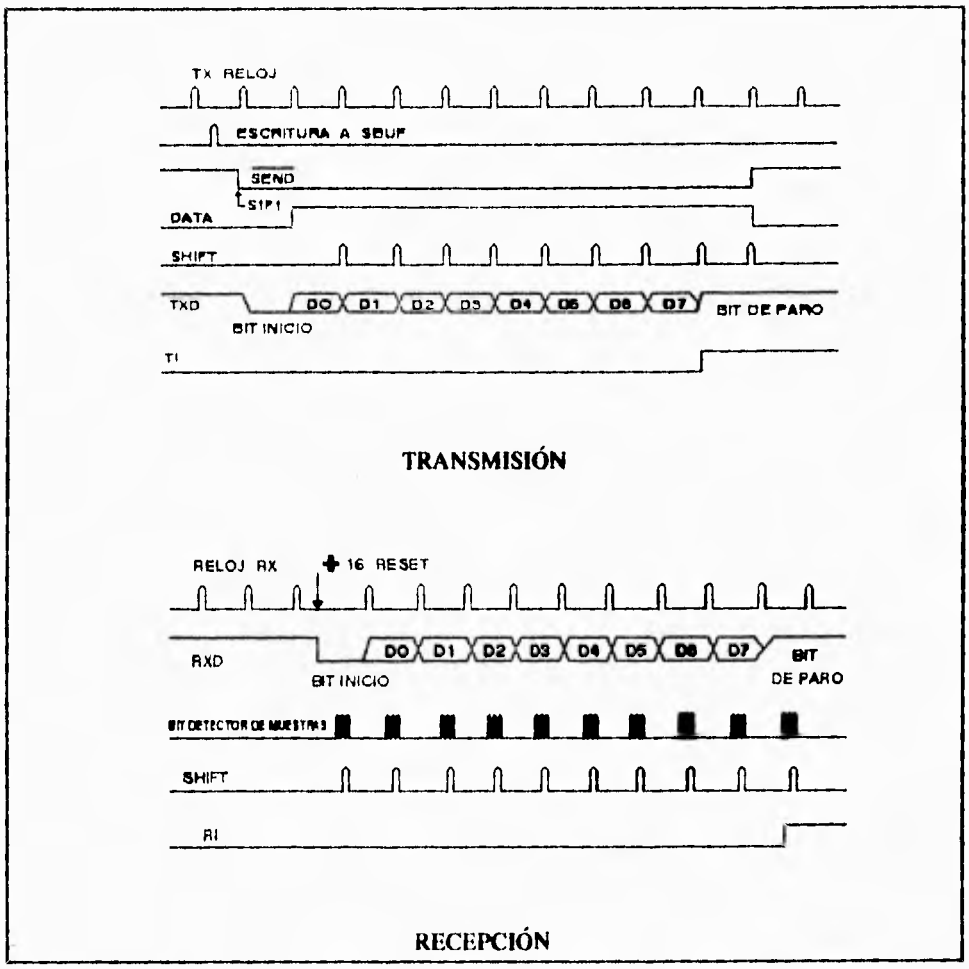


Figura 2.5. Puerto Serial en **Modo 1**.

Como los bits de datos vienen de la derecha, el primer bit se desplaza a la izquierda. Cuando el bit de inicio llega a la última posición del lado izquierdo del registro de corrimiento, (el cual en el modo 1 es un registro para el noveno bit), se activa una señal indicándole al bloque de Control de RX que realice un último corrimiento, carga SBUF y RB8, y activa RI. La señal para cargar SBUF y RB8, y activar RI, será generada si y sólo si, las siguientes condiciones están

presentes en el momento en el que el pulso final de corrimiento es generado.

1) RI = 0, y
2) Si SM2 = 0 o si el bit de paro recibido es igual a uno.

Si cualquiera de las dos condiciones no se presentan, la cadena recibida se pierde irremediablemente. Si ambas condiciones se presentan, el bit de paro está en RB8, los 8 bits de datos están en SBUF, y RI es activada. En este momento, ya sea las condiciones mencionadas estén presentes o no, la unidad regresa a buscar una transición de 1 a 0 en RXD.

INTERRUPCIONES

El 8751 provee cinco fuentes de interrupción: Interrupciones externas 0 y 1, interrupciones del Timer 0 y 1, y la interrupción del puerto serial.

Las interrupciones externas INT0 e INT1 (ambas negadas) pueden cada una de ellas activarse por nivel o por transición, dependiendo de los bits IT0 o IT1 en el registro TCON. De hecho las banderas que generan estas interrupciones son los bits IE0 e IE1 en TCON. Cuando una interrupción externa es generada, la bandera que se generó será limpiada por el hardware cuando la rutina de servicio es atendida esto solamente en el caso de que se active por transición. Si la interrupción fue activada por nivel, entonces la fuente externa que realizó la solicitud es la que controla la bandera solicitada, en vez del hardware del microcontrolador.

Las interrupciones del Timer 0 y el Timer 1 son generadas por TF0 y TF1, las cuales son habilitadas por un desplazamiento sucesivo en sus respectivos registros de Timer/Contador (excepto para el Timer 0 en el modo 3, ver la sección respectiva a los Timers/Contadores para mayor información). Cuando una interrupción de un timer es generada, la bandera que se generó será limpiada por el hardware del microcontrolador cuando la rutina de servicio haya sido atendida.

La interrupción del Puerto Serial es generada por la OR lógica de RI and TI. Ninguna de estas banderas es limpiada por hardware cuando la rutina de servicio es atendida. De hecho, la rutina de servicio normalmente va a tener que determinar si fue RI o TI quien generó la interrupción, y el bit deberá de ser limpiado por software.

Todos estos bits que generan interrupciones pueden ser activados o desactivados por software, con el mismo resultado aunque sean activadas o desactivadas por hardware. Esto es,

interrupciones pueden ser generadas o interrupciones pendientes pueden ser canceladas por software.

Cada una de estas fuentes de interrupción pueden ser habilitadas o deshabilitadas individualmente activando o desactivando un bit en el Registro de Función Especial IE. Se puede observar que IE contiene también un bit deshabilitador que es global, EA, el cual deshabilita todas las interrupciones.

(MSB)							(LSB)
EA	X	ET2	ES	ET1	EX1	ET0	EX0

IE: Registro de control para habilitación de interrupciones.

SÍMBOLO	POSICIÓN	FUNCIÓN
EA	IE.7	Deshabilita todas las interrupciones. Si EA=0 las interrupciones no serán reconocidas. Si EA=1, cada fuente de interrupción es individualmente habilitada o deshabilitada activando o desactivando su bit de habilitación.
-	IE.6	Reservado.
ET2	IE.5	Habilita o deshabilita la sobrecarga del Timer 2 o la interrupción de captura. Si ET2=0, la interrupción del Timer 2 es deshabilitada.
ES1	IE.4	Habilita o deshabilita la interrupción del Puerto Serial. Si ES=0, la interrupción del Puerto Serial es deshabilitada.
ET1	IE.3	Habilita o deshabilita la sobrecarga del Timer 1. Si ET1=0, la Interrupción del Timer 1 es deshabilitada.
EX1	IE.2	Habilita o deshabilita la Interrupción Externa 1. Si EX1=0, la Interrupción Externa 1 es deshabilitada.
ET0	IE.1	Habilita o deshabilita la sobrecarga del Timer 0. Si ETO=0, la Interrupción del Timer 0 es deshabilitada.
EX0	IE.0	Habilita o deshabilita la Interrupción Externa 0. Si EXO=0, la Interrupción Externa 0 es deshabilitada.

Estructura del Nivel de Prioridad.

Cada fuente de interrupción puede también ser programada individualmente por uno de dos niveles de prioridad activando o desactivando un bit del Registro de Función Especial IP. Una interrupción de baja prioridad puede por sí misma ser interrumpida por una interrupción de más alta prioridad, pero no por otra interrupción de más baja prioridad. Una interrupción de alta prioridad no puede ser interrumpida por ninguna otra fuente de interrupción.

(MSB)							(LSB)
X	X	PT2	PS	PT1	PX1	PT0	PX0

IP: Registro de control de prioridad de interrupciones.

SÍMBOLO	POSICIÓN	FUNCIÓN
-	IP.7	Reservado.
-	IP.6	Reservado.
PT2	IP.5	Define el nivel de prioridad de la interrupción del Timer 2. PT2 = 1 la programa con un nivel de prioridad alto.
PS	IP.4	Define el nivel de prioridad de la interrupción del Puerto Serial. PS = 1 la programa con un nivel de prioridad alto.
PT1	IP.3	Define el nivel de prioridad de la interrupción del Timer 1. PT1 = 1 la programa con un nivel de prioridad alto.
PX1	IP.2	Define el nivel de prioridad de la Interrupción Externa 1. PX1 = 1 la programa con un nivel de prioridad alto.
PT0	IP.1	Define el nivel de prioridad de la Interrupción del Timer 0. PT0 = 1 la programa con un nivel de prioridad alto.
PX0	IP.0	Define el nivel de prioridad de la Interrupción Externa 0. PX0 = 1 la programa con un nivel de prioridad alto.

Si dos peticiones de diferentes niveles de prioridad son recibidas simultáneamente, la petición de más alto nivel es atendida. Si peticiones del mismo nivel de prioridad son recibidas

simultáneamente, una secuencia de poleo externo determina cuál petición es atendida. Así dentro de cada nivel de prioridad hay una segunda estructura determinada por la secuencia de poleo, como sigue:

	Fuente	Prioridad dentro del Nivel
1.	IE0	Más Alta
2.	TF0	
3.	IE1	
4.	TF1	
5.	RI + TI	Más Baja

Se debe notar que la estructura "dentro del nivel de prioridad" es solamente usada para resolver peticiones simultáneas dentro del mismo nivel de prioridad.

Manejo de interrupciones.

Las banderas de interrupción son muestreadas en S5P2 de cada ciclo de máquina. Las muestras son poleadas durante los subsiguientes ciclos de máquina. Si una de las banderas está activada en S5P2 del ciclo anterior, el ciclo de poleo la encontrará y el sistema de interrupción generará un LCALL a la rutina de servicio apropiada, permitiendo que la llamada LCALL generada por hardware no sea bloqueada por ninguna de las siguientes condiciones:

1. Si una interrupción de igual o elevada prioridad ya está en progreso.
2. El actual (poleo) ciclo no es el ciclo final en la ejecución de la instrucción en progreso.
3. La instrucción en progreso es RETI o cualquier acceso de los registros IE o IP.

Cualquiera de estas tres condiciones bloqueará la generación de LCALL a la rutina de servicio de interrupción. La condición 2 asegura que la instrucción en progreso será completada antes de atender cualquier rutina de servicio. La condición 3 asegura que si la instrucción en progreso es RETI o cualquier acceso a IE o IP, entonces al menos una instrucción más será ejecutada antes de atender cualquier interrupción.

El ciclo de sondeo es repetido con cada ciclo de máquina, y los valores sondeados serán los valores que fueron presentados en S5P2 en el ciclo de máquina previo. Por lo tanto si una bandera de interrupción es activada pero no está por encima de una de las condiciones, en caso

de que la bandera no continué activa cuando la condición de bloqueo es removida, la interrupción negada no será atendida. En otras palabras, el hecho de que la bandera de interrupción estuvo activa y no fue atendida implica que no será recordada. Cada ciclo de sondeo es nuevo.

Así el procesador reconoce una petición de interrupción para que se genere una llamada LCALL vía hardware, a la apropiada rutina de servicio. En algunos casos también limpia las banderas que generaron la interrupción, y en otros casos no lo hace. Nunca limpia las banderas del Puerto Serial y del Timer 2. Esto lo debe realizar el usuario por medio de software. Limpia la bandera de interrupción externa (IE0 o IE1) solamente si fue activada por medio de una transición. La generación de LCALL por medio de hardware salva los contenidos del contador de programa dentro del stack (pero no salva el PSW) y recarga el PC con una dirección que depende de la fuente de interrupción que ha sido solicitada, como se muestra a continuación:

Fuente	Dirección
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI+TI	0023H

La ejecución de la interrupción se lleva cabo al encontrar la localidad de memoria correspondiente, finalizando ésta, al encontrar la instrucción RETI. Es entonces que, la instrucción RETI informa al procesador que esta rutina de interrupción no está en progreso, saca los dos bytes que se encuentran en la parte superior del stack y recarga el Contador de Programa. La ejecución del programa interrumpido continúa en donde se había quedado.

Se debe notar que una simple instrucción RET también retornará la ejecución al programa interrumpido, pero podría dejar pensando al sistema de control de interrupción que alguna interrupción continúa en progreso.

Interrupciones Externas.

Las fuentes externas pueden ser programadas para ser activadas por nivel o por transición al habilitar o deshabilitar el bit IT1 o IT0 en el registro TCON. Si ITx = 0, la interrupción externa x es activada al detectar un nivel bajo en la terminal INTx (negado). Si ITx = 1, la interrupción externa x es activada por flanco. En este modo si sucesivas muestras de la terminal

INTx muestran un nivel alto en un ciclo y un nivel bajo en el siguiente ciclo, la bandera de petición de interrupción IEx en TCON es habilitada. La bandera del bit IEx entonces solicita la interrupción.

Después de que las terminales de la interrupción externa son muestreados una vez cada ciclo de máquina, una entrada alta o baja debe retenerse por lo menos 12 períodos de oscilación para asegurar el muestreo. Si la interrupción externa es activada por transición, la fuente externa debe retener la terminal de petición alto al menos un ciclo (12 períodos de oscilación), y entonces retenerlo bajo por lo menos un ciclo para asegurar que la transición es vista de tal manera que la bandera de petición de interrupción será habilitada. IEx será automáticamente limpiada por el CPU cuando la rutina de servicio es llamada.

Si la interrupción externa es activada por nivel, la fuente externa debe retener la petición activada hasta que la interrupción solicitada sea en realidad generada. Entonces debe desactivarse la petición antes de que la rutina de servicio de la interrupción sea completada, o de lo contrario otra interrupción será generada.

RESET.

La inicialización de las entradas (reset input) se realiza con la terminal RST, el cual es la entrada de un circuito biestable (disparador).
Un reset se lleva a cabo al mantener la terminal RST en alto por lo menos dos ciclos de máquina (24 períodos de oscilación), mientras el oscilador está corriendo. El CPU responde, ejecutando un reset interno. También configura las terminales del ALE y del PSEN (negado) como entradas. (Ellos son casi bidireccionales). El reset interno es ejecutado durante el segundo ciclo en el cual RST es alto y es repetido cada ciclo de máquina hasta que RST cambie a bajo. Esto deja los registros internos como sigue:

REGISTROS	CONTENIDO
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	0FFH
IP	(XX000000)
IE	(0X000000)
TMOD	00H

TCON	00H
T2CON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	INDETERMINADO
PCON	(0XXX0000)

La RAM interna no es afectada por el reset. Cuando VCC es encendido, el contenido de la RAM es indeterminado a menos que el dispositivo esté retornando de la operación de bajo consumo de energía.

(MSB)							(LSB)
SMOD	-	-	-	GF1	GF0	PD	-

PCON: REGISTRO DE CONTROL DE ENCENDIDO

SÍMBOLO	POSICIÓN	NOMBRE Y FUNCIÓN
SMOD	PCON.7	Bit doblador de la velocidad de transmisión. Cuando es colocado a uno, la velocidad de transmisión será doblada si el puerto serial es usado en cualquiera de los modos 1, 2 o 3.

SÍMBOLO	POSICIÓN	NOMBRE Y FUNCIÓN
-	PCON.6	Reservado.
-	PCON.5	Reservado.
-	PCON.4	Reservado.
GF1	PCON.3	Bandera de propósito general.
GF0	PCON.2	Bandera de propósito general.
PD	PCON.1	Bit de reducción de energía. Activando este bit se activa la operación de bajo consumo de energía.
-	PCON.0	No está disponible en el 8751H.

CAPITULO 3

PLANTEAMIENTO DEL PROBLEMA

3. PLANTEAMIENTO DEL PROBLEMA.

Las operaciones manuales en fabricación, supervisión o exploración que se caractericen como inseguras o arriesgadas para los trabajadores humanos que las realizan, han sido tradicionalmente las situaciones ideales para aplicaciones de robots; y esta característica se presenta en una planta nuclear en el momento en el que se realizan trabajos de inspección o de reparación donde los riesgos se incrementan notablemente para la salud del personal técnico; por lo que una de las funciones que realizará la Sonda Robot desarrollada en este trabajo es precisamente la de inspección.

La Sonda se divide en cuatro etapas, como son la etapa de control de movimiento en Tiempo Real de un manipulador de seis grados de libertad (como máximo), la etapa de tracción, la de adquisición de datos y la de comunicación. Adicionalmente se cuenta con la interfase necesaria para comunicarse con una computadora personal para generación de tareas, las cuales puedan ser utilizadas en el interior del reactor, esto con el fin de contar con un sistema de aprendizaje. El sistema se desarrolla en base a un microcontrolador de la familia MCS-51 (8751H) como elemento principal, pues es en este dispositivo donde se almacena el programa monitor que controla todos los dispositivos periféricos, así como las rutinas necesarias para realizar la comunicación con la computadora.

Etapa de Control

El Manipulador realizará rutinas de muestreo lo que permitirá monitorear un área de trabajo. El control se realiza con la interfase periférica programable (82C55), conectada al microcontrolador. La operación podrá realizarse desde una computadora personal.

Etapa de Tracción

Tiene la función de proporcionar movimiento a la Sonda, de tal manera que permita al operador posicionarla en el lugar deseado: Consta de dos motores de DC, uno para la tracción y otro para la dirección los cuales también son controlados con el 82C55.

Etapa de Adquisición de Datos

En esta etapa se conectarán los sensores para la adquisición de los parámetros críticos de la planta (máximo 16) como son radiación, temperatura y humedad relativa, así como la detección de obstáculos (diseño a futuro). Éste modulo incluirá un medio de almacenamiento temporal para preservar las lecturas tomadas.

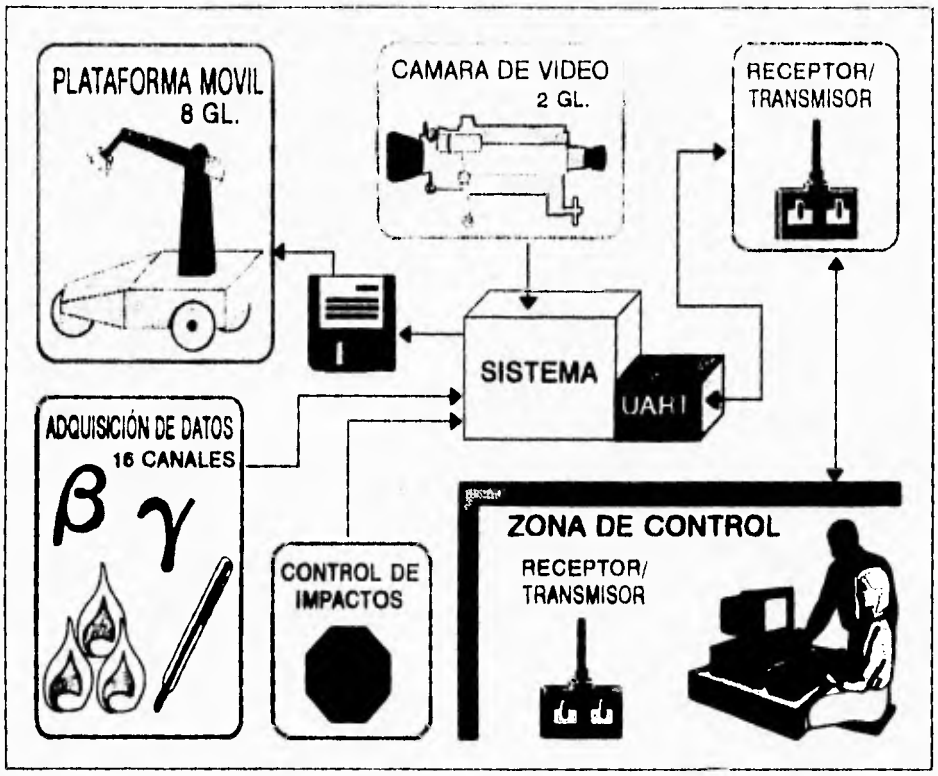


Diagrama de Bloques de la Sonda Robot.

Etapa de Comunicación

El propósito de éste módulo es el que la Sonda cuente con un canal de comunicación de doble vía con el operador: el primero se necesita para que reciba las órdenes del operador y el segundo para transmitir la información adquirida para ser analizada.

CAPITULO 4

DESARROLLO Y CONSTRUCCIÓN

En éste capítulo se explican los aspectos y criterios que se tomaron en cuenta para el diseño de la Sonda Robot.

4. DESARROLLO Y CONSTRUCCIÓN.

En la Figura 4.1, se muestra la arquitectura del sistema en bloques, en la cual se observa en forma global los componentes principales, la parte más importante del sistema es el microcontrolador (8751H) que es el cerebro de éste, y el lugar en donde es grabado el programa que controla todos los periféricos del sistema; así como, las instrucciones necesarias para realizar la comunicación con una computadora personal y con el Sistema de Control Remoto.

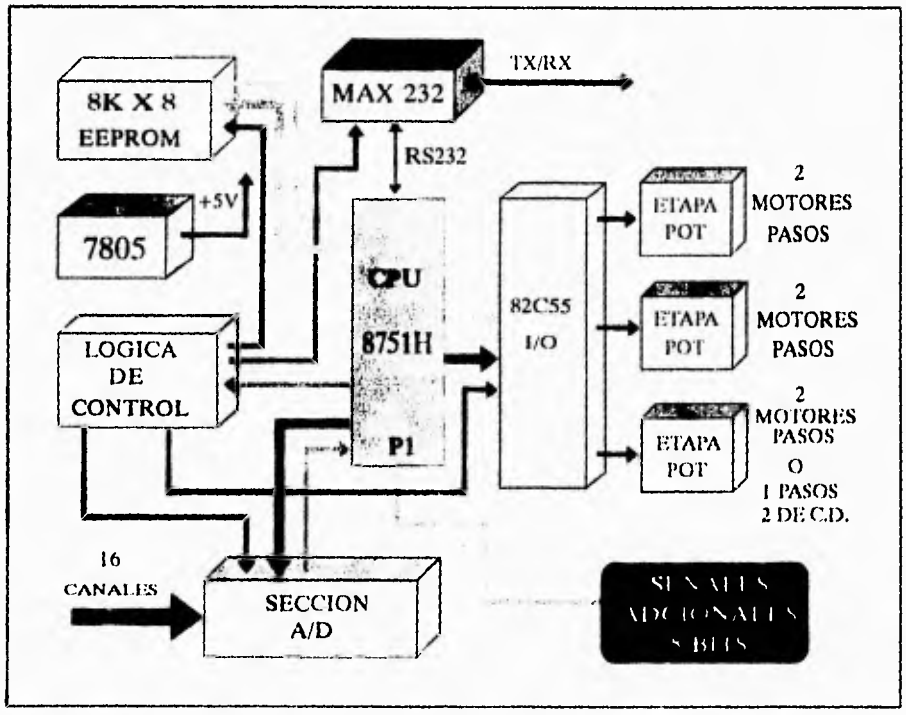


Figura 4.1. Diagrama de Bloques del Sistema.

El sistema se puede dividir en cuatro etapas fundamentales que son: la etapa de comunicación serial, la de control del brazo mecánico, la de tracción y el sistema de adquisición de datos.

Hablando del Sistema completo, la operación se está realizando en tiempo real, puesto que es indispensable tener movimiento inmediato en el Brazo Mecánico, así como en la tracción. El tipo de brazo que se está manejando es de coordenadas de revolución o articuladas (tres ejes rotacionales) con un máximo de 6 grados de libertad, construido por Electrónica Veneta.

El control de todos los periféricos se realiza por medio de las interrupciones del 8751H, el cual provee cinco fuentes de interrupción: Interrupciones externas IE0 e IE1, interrupciones del Reloj TF0 y TF1, y la interrupción del puerto serial generada por la OR lógica de RI y TI.

Pueden ser programadas dentro de uno de dos niveles de prioridad, con la ayuda del registro IP del microcontrolador (Registro de control de prioridad de interrupciones). En el caso en el que dos peticiones de diferentes niveles de prioridad sean recibidas simultáneamente, la petición de más alto nivel es atendida. Si peticiones del mismo nivel de prioridad son recibidas simultáneamente, la petición atendida será la que tenga mayor jerarquía dentro del nivel, el cual está definido de la siguiente manera:

	FUENTE	Prioridad dentro del Nivel
1.	IE0	Más Alta
2.	TF0	
3.	IE1	
4.	TF1	
5.	RI+TI	Más Baja

Se debe observar que la estructura "dentro del nivel de prioridad" es solamente usada para resolver peticiones simultáneas dentro del mismo nivel de prioridad.

Para el Sistema quedó definida la estructura de prioridad de la siguiente forma:

Fuente		Prioridad
IE0	Nivel 2	Más Alta
TF0	Nivel 2	
IE1	Nivel 2	
RI + TI	Nivel 2	
TF1	Nivel 1	Más Baja

El control del Brazo Mecánico y de la Tracción se realizan por medio de la Interrupción Serie, al activarse el bit de Interrupción RI en el registro SCON (Registro de Control del Puerto Serial). Al ser atendida la petición de interrupción son codificadas las señales (provenientes de la PC o del Sistema de Control Remoto según sea el caso) para realizar la tarea deseada.

La Transmisión de Datos, se realiza también por medio de la Interrupción Serie, pero ésta se genera al activarse el bit TI (con un uno lógico), también presente en SCON.

Como se mencionó anteriormente la Interrupción Serie es generada por la OR lógica de RI y TI por lo que es necesario incluir una rutina en el programa para poder determinar cuál de las dos fuentes de interrupción realizó la petición, y que sea atendida. En caso de que se generen las dos simultáneamente, la rutina atenderá primero a la Interrupción generada por la bandera de Transmisión y posteriormente a la de Recepción.

Para la comunicación entre la Etapa de Adquisición de Datos y el 8751H, se utiliza la interrupción externa IE1; la cual es programada para activarse al presentarse un flanco de bajada en la terminal INT 1 (negado), para lograr la sincronización de señales entre el 8751H y el ADC0817, como se explicará más adelante.

La Interrupción del Reloj 0 (TF0), la cual es activada al presentarse una transición de 1 a 0 en la terminal externa T0, es utilizada para llevar el conteo de la señal enviada por el contador de vueltas (tacómetro) que forma parte del Sistema de Localización.

Como se ve en el Diagrama Eléctrico 1, el sistema cuenta con una interfase periférica programable, la cual fue necesaria añadir, con el fin de expandir las capacidades del sistema, pues es necesario contar con otro puerto paralelo. Se eligió el circuito 82C55 de INTEL por ser de la misma familia que el microcontrolador, lo que los hace completamente compatibles; además de que este dispositivo cuenta con tres puertos paralelos, lo que otorga un total de siete puertos (como se verá más adelante son necesarios dos puertos del 8751H para controlar al 82C55).

También es utilizado un convertidor analógico-digital, por lo que se añadió al sistema el ADC0817 de 8 bits y 16 Canales, compatible completamente con el microcontrolador.

El diseño cuenta con un medio de almacenamiento de datos, para tener un respaldo de los mismos, con lo que se integra una EEPROM; que por ser eléctricamente grabable y borrable nos permite el almacenamiento de los datos, inmediatamente después de ser adquiridos por el ADC0817.

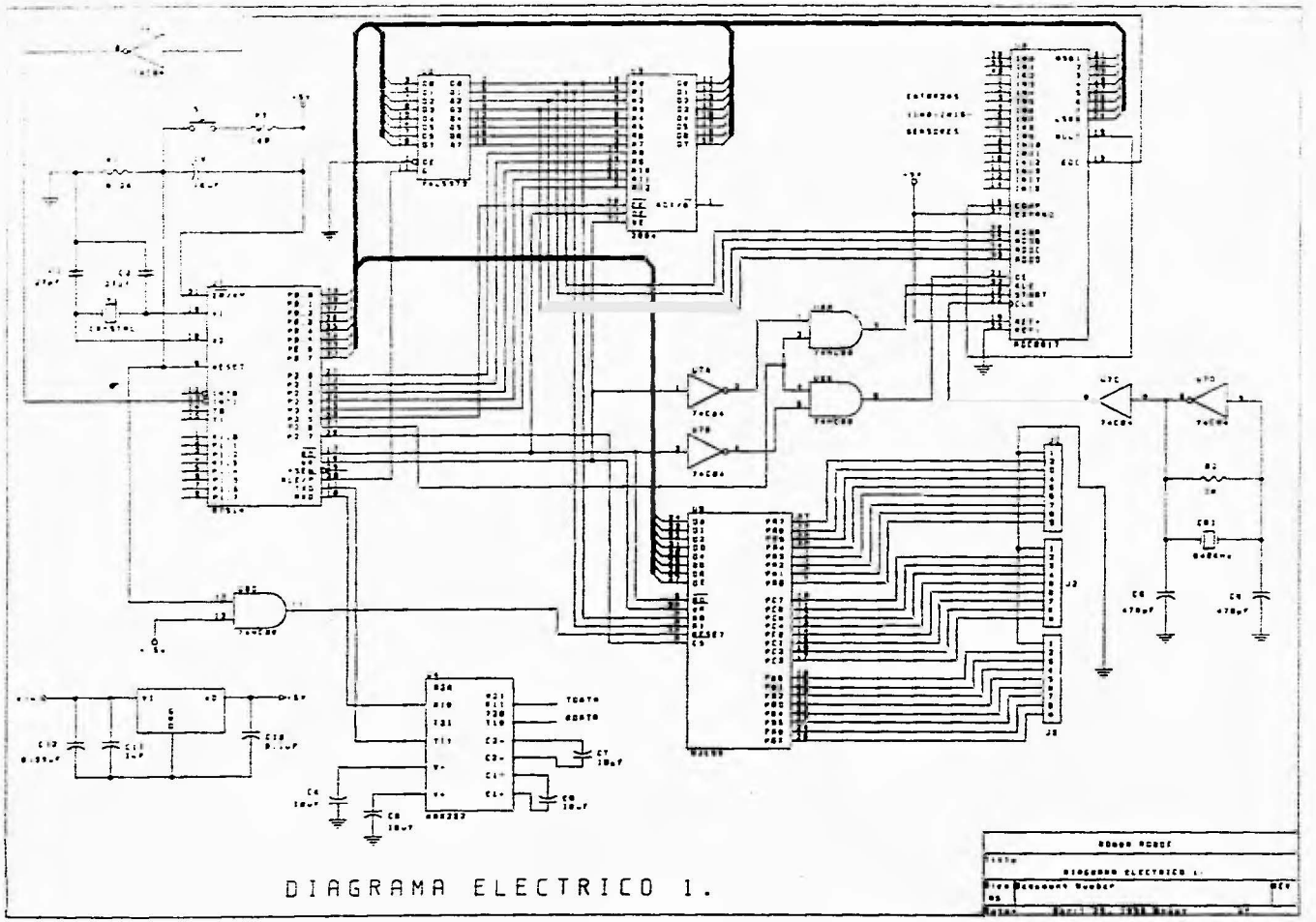


DIAGRAMA ELECTRICO 1.

La activación de la EEPROM (U3), del ADC0817 (U6) y del 82C55 (U5), se efectúa al realizar accesos a dispositivos externos, función realizada por medio de los puertos P0 y P2 del 8751H, donde son multiplexadas las direcciones y los datos.

Los accesos son realizados en este caso con los 16 bits de direcciones (MOVX @dptr), por lo que P0 y P2 no pueden ser usados para propósito general de entrada/salida. Los 8 bits más altos de direcciones se encuentran en el Puerto 2, los cuales son retenidos durante el tiempo de duración del ciclo de lectura o escritura; mientras que las direcciones bajas y los datos son multiplexados en tiempo en el Puerto 0, en este tipo de aplicación el Puerto 0 no tiene salidas de colector abierto por lo que no requiere resistencias externas para realizar esta función. La señal ALE (Habilitador del "latch" de direcciones) debe ser usada para capturar el byte de direcciones en un "latch" externo (74LS373 U2). La siguiente figura muestra el mapa de memoria del sistema.

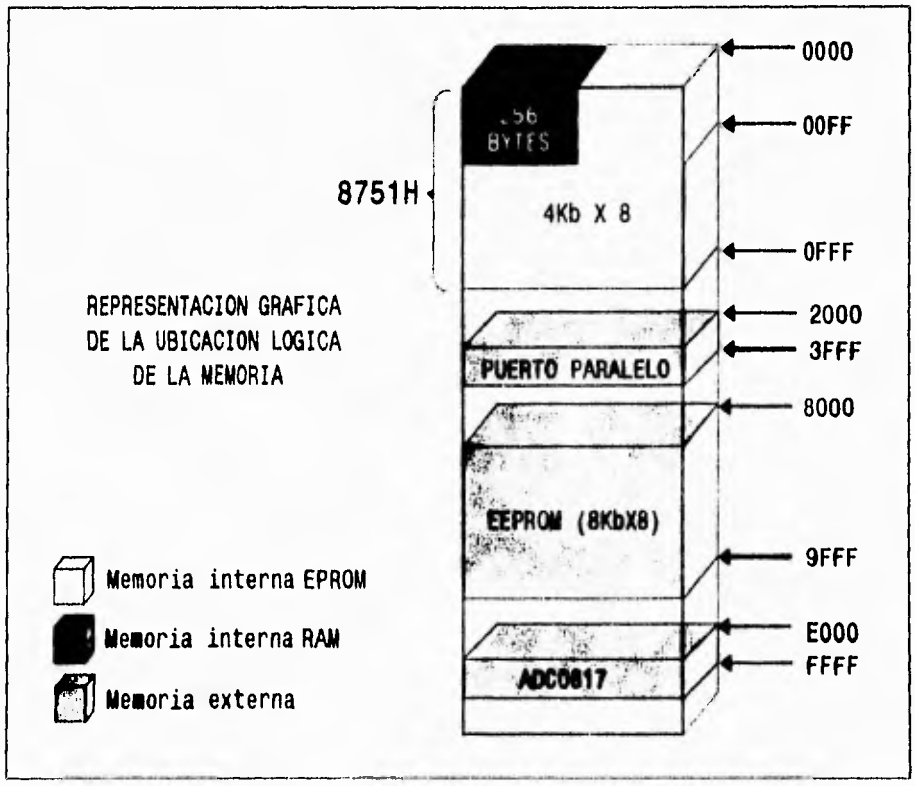


Figura 4.2a. Mapa de Memoria del Sistema.

A continuación se presenta el mapa de memoria externa, que muestra la dirección de cada uno de los periféricos.

	BUS DE DIRECCIONES																
	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	HEX
82C55	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
EEPROM	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	9FFF
LIBRE	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFF
ADC0817	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

Líneas de Control

BUS DE DATOS

BUS DE DIRECCIONES	82C55	
2000 H	PUERTO A	MOTOR 1 LSB MOTOR 2 MSB
2001 H	PUERTO B	MOTOR 3 LSB MOTOR 4 MSB
2002 H	PUERTO C	MOTOR 5 LSB MOTOR 6 MSB
2003 H	PROGRAMACIÓN	

BUS DE DIRECCIONES	ADC0817
E000 H	CANAL 1
E001 H	CANAL 2
E002 H	CANAL 3
⋮	⋮
E00E H	CANAL 15
E00F H	CANAL 16

Figura 4.2b. Mapeo de Memoria Externa.

4.1 Etapa de Comunicación.

El 8751H cuenta con 4 puertos bidireccionales, usándose el canal serial que es parte del puerto tres para realizar la comunicación con el puerto serie de la PC y con el Sistema de Control Remoto.

El canal serial esta configurado como un UART de 8 bits (Transmisor-Receptor Universal Asíncrono) en Modo 1. En este modo es posible variar la velocidad de transmisión, tendiéndose 10 bits de los cuales: 8 bits son de datos, 1 bit de inicio y uno de paro. La velocidad de transmisión es determinado por el Reloj 1 en el modo de auto recarga (Modo 2) y operándolo como reloj; por lo tanto la parte alta del registro TMOD = 0010B. En este caso, la velocidad de transmisión es obtenida por medio de los 4 bits más significativos del Reloj 1 (TH1), el cual es obtenido por medio de la siguiente fórmula:

$$TH1 = \frac{(BR)(98304) - 2^{SMOD}\ (F_{osc})}{384(BR)}$$

Donde:

- TH1 es cargado en hexadecimal.

- BR es la Velocidad de Transmisión a la que se quiera transmitir/recibir.

- SMOD es el bit número 7 del registro PCON (Registro de control de encendido) utilizado para doblar la Velocidad de Transmisión. Cuando SMOD = 1, la Velocidad de Transmisión es doblada si el puerto serial es usado en cualquiera de los Modos 1, 2 o 3.

Para este caso SMOD = 0, F_{osc} = 6 MHz y BR = 1200 Bauds, con lo que se obtuvo que TH1 = F3H. *En la Tabla 1 del capítulo dos, se muestran valores calculados para diferentes condiciones, mientras que la Figura 4.3 muestran el funcionamiento del UART en el Modo 1.*

Una vez configurado el canal serial a la Velocidad de Transmisión deseada, se requiere circuitería adicional para realizar la comunicación con el puerto serial de la PC, con lo que es utilizado el MAX232 (U4) para este propósito, ya que cuenta con un circuito transmisor/receptor doble, y un cambiador de nivel TTL a CMOS; que son los niveles utilizados por el 8751H y la PC respectivamente. Es un circuito ideal para simplificar el diseño de las fuentes de poder, al funcionar únicamente con 5 volts, además de cumplir con el estándar de comunicación ANSI/EIA-RS-232. La Figura 4.4 muestra el diagrama de bloques de toda la etapa de comunicación, mientras que el Diagrama Eléctrico 2, muestra todos los componentes electrónicos de la misma.

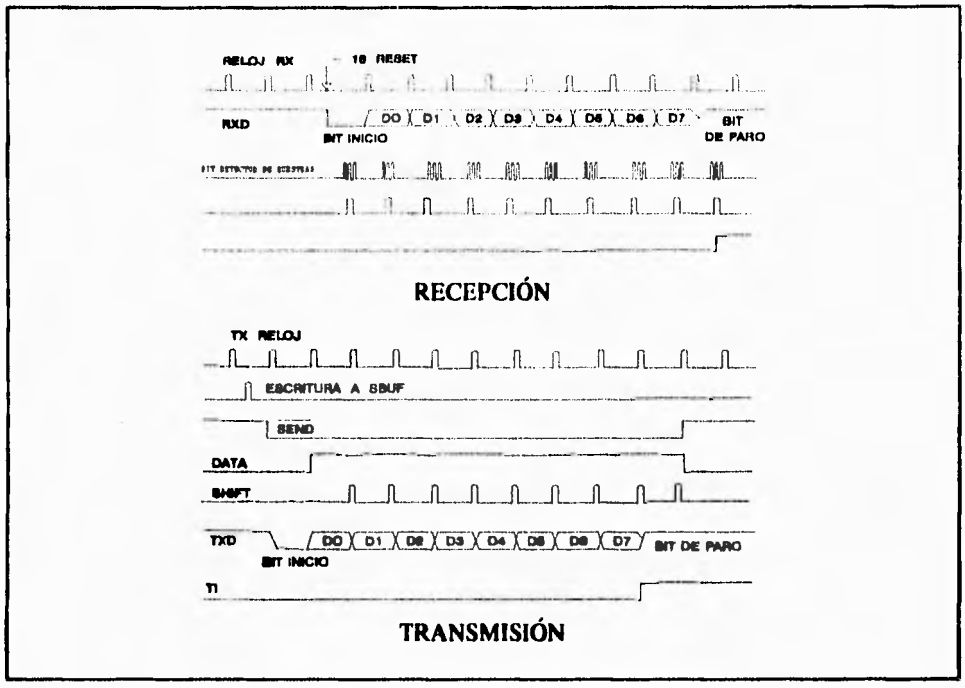


Figura 4.3. Puerto Serial Modo 1.

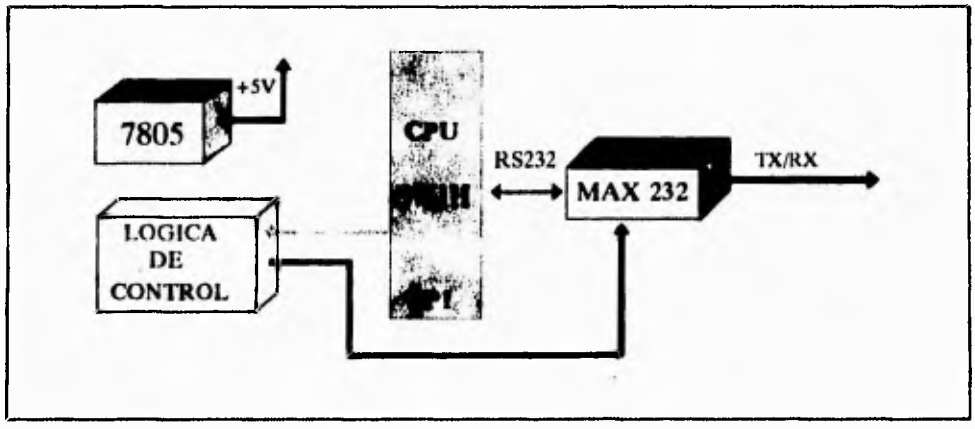


Figura 4.4. Diagrama de bloques de la Etapa de Comunicación Serial.

El MAX232 está dividido en tres secciones: un transmisor doble, un receptor doble y un convertidor doble de voltaje para elevar el voltaje de +5V a +/- 10V. La sección del convertidor tiene dos elevadores de carga, la primera utiliza un capacitor externo C6 (Diagrama Eléctrico 2) para duplicar los +5V de entrada a +10V, con una impedancia de salida de aproximadamente 200 ohms. La segunda emplea el capacitor externo C7 como se ve en el Diagrama Eléctrico 2 para invertir los +10V a -10V, con una impedancia de salida total de 450 ohms (incluyendo los efectos de la impedancia del doblador de voltaje).

El valor utilizado para los capacitores C4 a C7 es de 10 microfaradios. Aunque el valor de los capacitores no es crítico, si se incrementa el valor de C6 y C7, disminuirá la impedancia de salida del doblador y del inversor. Incrementando el valor de C4 y C5 disminuye el voltaje de rizo de las fuentes, por lo que disminuye la frecuencia en las salidas RS-232.

Cada uno de los dos transmisores es un inversor CMOS, encendido por los +/- 10V de entrada generados internamente por las fuentes de alimentación. La entrada es compatible con TTL y CMOS, con un umbral de aproximadamente el 26 % de Vcc (1.3 V para 5V Vcc). Si alguna de las entradas del trasmisor no es utilizada, puede dejarse desconectada, ya que contiene una resistencia interna de colector abierto (pull-up) de 400 Kilohms, conectada entre la entrada del trasmisor y Vcc, fijando la entrada en alto y la salida no utilizada en bajo.

Los dos receptores cumplen completamente las especificaciones RS-232: la impedancia de entrada se sitúa entre 3 y 7 Kilohms, pueden soportar más de +/- 30 volts de entrada (con o sin voltaje aplicado de 5V) y los umbrales de entrada se encuentran dentro de los límites de +/- 3 volts de las especificaciones RS-232

Para asegurar la compatibilidad con entradas RS-232 o TTL/CMOS, los receptores del MAX232 tiene un V_{IL} de 0.8V y un V_{IH} de 2.4 V. Los receptores tienen 0.5 volts de histéresis para mejorar el rechazo al ruido. Las salidas compatibles TTL/CMOS son bajas cuando la entrada RS-232 sea mayor de 2.4V, y la salida del receptor será alta cuando la entrada esté en alta impedancia o manejándose entre +0.8V y -30V.

El microcontrolador (U1) tiene integrada una memoria EPROM de 4 KBytes x 8 en donde se grabo el programa de control; éste es realizado en lenguaje ensamblador, utilizando un editor de texto, un ensamblador de la familia MCS-51 y un simulador para la realización de pruebas del programa.

El circuito U1 tiene conectado un circuito de reloj a X1 (Entrada) y X2 (Salida), que consiste en dos capacitores cerámicos (C1 y C2) y un cristal (Y1) de 6 MHz, que es a la frecuencia de oscilación a la que opera el circuito.

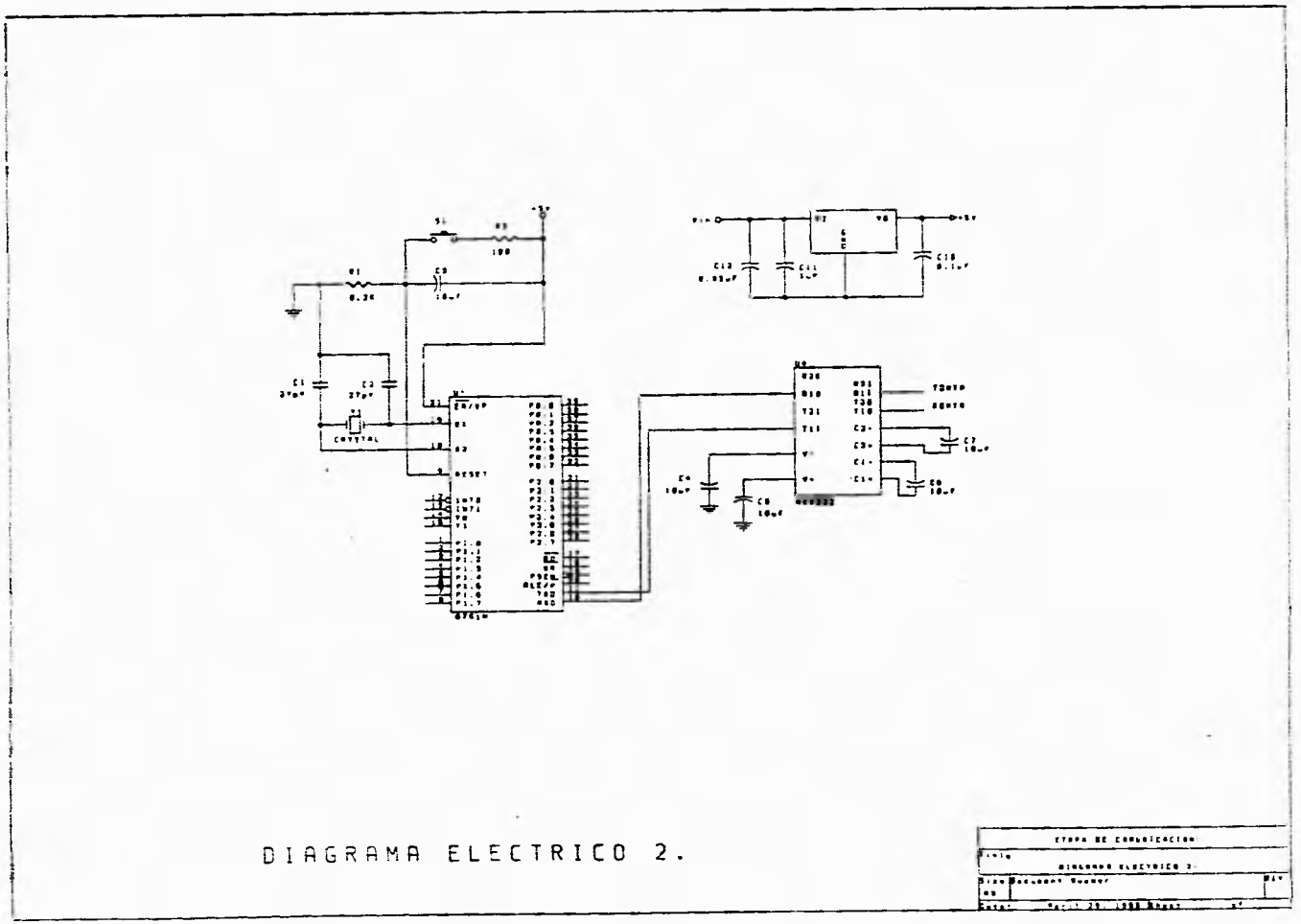


DIAGRAMA ELECTRICO 2.

4.2 CONTROLADOR DE MOVIMIENTO PARA EL MANIPULADOR y LA TRACCIÓN.

El control del manipulador se realiza por medio de la interfase periférica programable PPI 82C55 que cuenta con tres puertos y de una etapa de potencia, por medio de los cuales se operan hasta seis motores de pasos de un manipulador o bien cinco motores de pasos dejando libres cuatro bits para el control del desplazamiento (dirección y tracción) de la sonda como se muestra en el Diagrama de Bloques de la figura 4.5, mientras que en el Diagrama Eléctrico 3 se observan los componentes utilizados en esta etapa y su interconexión.

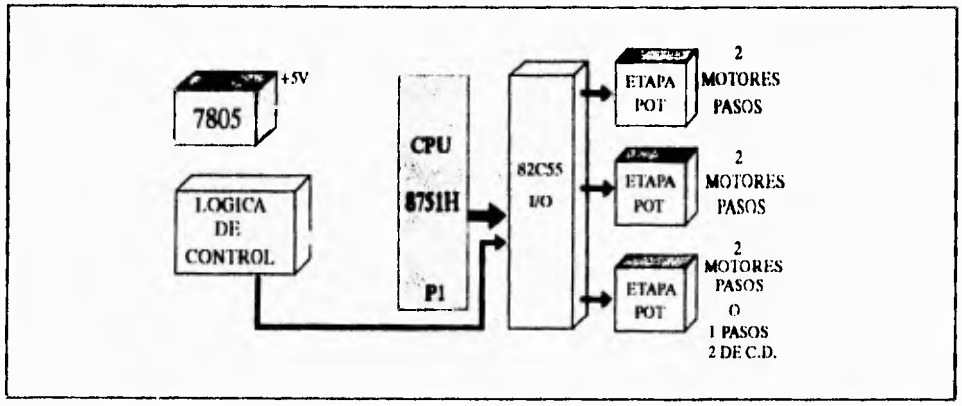


Figura 4.5 Diagrama de Bloques de la Etapa de Control para el Manipulador y el Sistema de Tracción.

El 82C55 tiene la capacidad de manejar en todas las salidas de los puertos hasta 2.5 mA de DC, señal que es utilizada como entrada para la etapa de potencia.

Las señales A1 y A0 en conjunto con RD y WR controlan la activación de uno de los tres puertos del 82C55 o del registro de palabra de control utilizado para su programación en cualquiera de los tres modos de operación, como se muestra a continuación:

A1	A0	RD	WR	CS	Operación de entrada (Lectura)
0	0	0	1	0	Puerto A - Bus de datos
0	1	0	1	0	Puerto B - Bus de datos
1	0	0	1	0	Puerto C - Bus de datos
1	1	0	1	0	Palabra de control - Bus de datos

A1	A0	RD	WR	CS	Operación de salida (Escritura)
0	0	1	0	0	Bus de datos - Puerto A
0	1	1	0	0	Bus de datos - Puerto B
1	0	1	0	0	Bus de datos - Puerto C
1	1	1	0	0	Bus de datos - Control

El 82C55 puede ser programado en tres modos de operación:

Modo 0: Entrada/salida
Modo 1: Protocolo de comunicación para entrada/salida
Modo 2: Bus bidireccional

La selección del modo de operación y la configuración de los puertos se realiza por medio de la palabra de control, como se muestra a continuación:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

donde D_0 Puerto C (bajo)
1 = entrada
0 = salida

D_1 Puerto B
1 = entrada
0 = salida

D_2 Selección de Modo para Puerto C (parte baja) y Puerto B
0 = Modo 0
1 = Modo 1

D_3 Puerto C (alto)
1 = entrada
0 = salida

D_4 Puerto A
1 = entrada
0 = salida

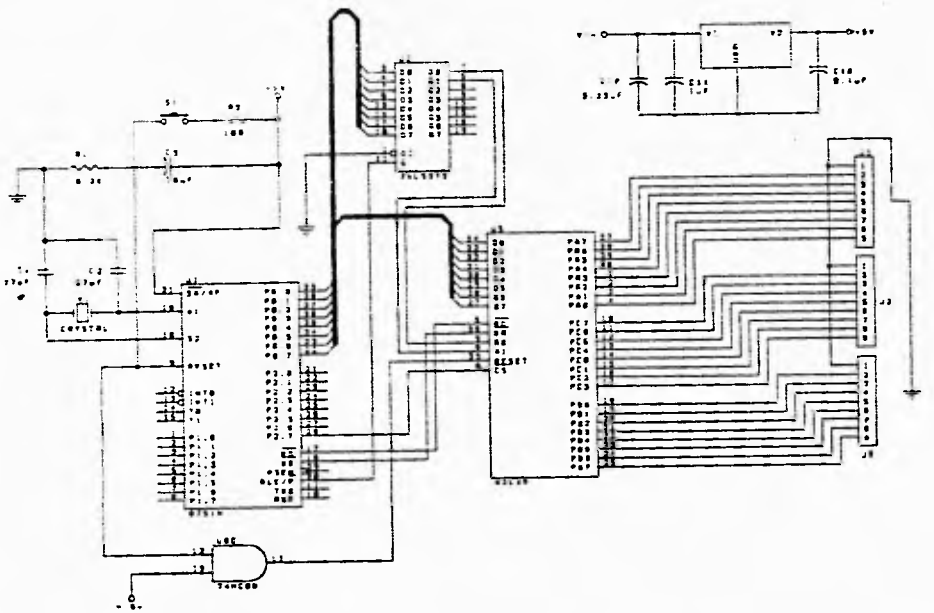


DIAGRAMA ELECTRICO 3.

D_6D_5 Selección de modo para Puerto C (parte alta) y Puerto A
00 = Modo 0
01 = Modo 1
1X = modo 2

D_7 Bandera de activación de modo
1 = activa

Los modos para el Puerto A y Puerto B, pueden ser definidos por separado, mientras que el Puerto C es dividido en dos porciones, definiéndose según la configuración del Puerto A y el Puerto B.

Para este caso se programó en Modo 0, con las 24 líneas de los puertos como salidas, es decir: Palabra de Control = 80 H.

La Figura 4.6 muestra los diagramas de tiempos del 8751H y del 82C55.

La habilitación del 82C55 se realiza con P2.7 (U1) en bajo, como se ve en el mapeo de memoria externa Figuras 4.2a y 4.2b.

De las figuras 4.2a y b se observa que una de las direcciones que es válida para el Puerto A del 82C55 es la 2000H, que es un puerto de 8 bits, con lo que es posible controlar dos motores de pasos de 4 fases cada uno.

Los 4 bits menos significativos del Puerto A controlan el Motor 1, mientras que los 4 bits más significativos controlan el Motor 2. Los otros dos puertos disponibles pueden controlar dos motores de pasos de 4 fases cada uno, al igual que el Puerto A. El Puerto C puede ser utilizado para manejar 1 motor de pasos en lugar de dos, con lo que se dejan libres cuatro bits para poder controlar dos motores de DC para la tracción de la Sonda.

Conociendo las direcciones de cada uno de los motores de pasos y la secuencia de pasos que hay que enviarles para producir un desplazamiento según se vio en el capítulo dos, sólo se necesita programar en la EPROM del 8751H (U1) la rutina de control para cada motor, rutina que se explica a continuación para el caso del motor de pasos 1, con movimiento hacia adelante.

MOV DPTR,#2000H	Se carga la dirección del Puerto A en el registro DPTR de 16 bits
MOV A,#0CH	Se carga el primer dato de la secuencia de pasos hacia adelante en el Acumulador

MOVX @DPTR,A	Se realiza una operación de escritura a un dispositivo externo (MOVX), multiplexando las direcciones y los datos
MOV A,#09H **MOVX @DPTR,A**	Segundo dato de la secuencia de pasos
MOV A,#03H **MOVX @DPTR,A**	Tercer dato de la secuencia de pasos
MOV A,#06H **MOVX @DPTR,A**	Último dato de la secuencia de pasos

Si se desea mover la articulación a un lugar específico, solamente es necesario ejecutar repetidamente esta secuencia hasta llegar a la posición deseada.

Por el contrario si se requiere regresar a la posición de origen, es decir, movernos hacia atrás, sólo se invierte la secuencia de datos para lograrlo.

Lo anterior es aplicable a los motores de pasos de los Puertos B y C del 82C55, tomando en cuenta que lo único que se cambia es la dirección del puerto.

En el programa de control las rutinas que ejecutan los movimientos del manipulador están identificadas con la etiqueta RUTINAS DE MOVIMIENTO, indicándose en cada caso el código con el cual se mueve la articulación, así como su dirección.

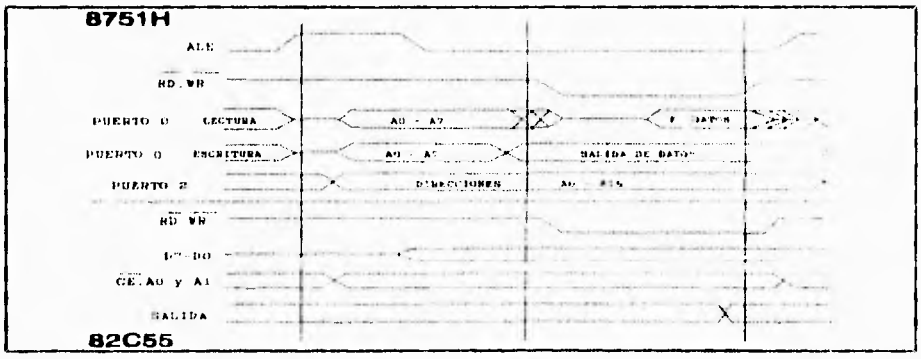


Figura 4.6. Diagramas de Tiempos del 8751H y del 82C55.

Se presenta un programa de control con el Lenguaje de Programación Pascal, por medio del cual es posible controlar los movimientos del Brazo desde la PC, teniendo la posibilidad de generación automática de tareas, que en este caso es posible debido al control en tiempo real que se realiza ya que la interacción de la PC con el sistema es directo.

La utilización de Pascal, se fundamenta en el hecho de ser un lenguaje de alto nivel, con lo que es posible interactuar con el BIOS de la PC (para manejar las interrupciones), al igual que con sus periféricos; contando adicionalmente con la paquetería disponible en la PC, en caso de ser necesarias.

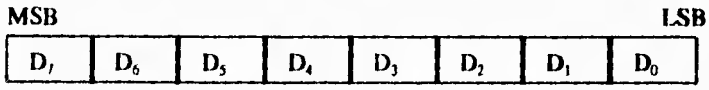
El programa en Pascal tiene la función de activar el puerto serial, y configurado para realizar la comunicación con la Sonda Robot utilizando el protocolo RS-232 para transmitir/recibir datos a 1200 Bauds también en tiempo Real. El programa permite mover el Brazo por medio del teclado, teniendo las opciones de grabar rutinas de control, utilizar rutinas previamente grabadas y paros de emergencia en caso de proximidad de impacto con algún objeto, así como, el control del desplazamiento de la sonda. Todas las señales de control son recibidas a través del UART de U1, donde son codificadas para luego ser enviada la secuencia de pasos correspondiente a cada motor vía los puertos del 82C55.

La configuración del puerto serial de la PC, se realiza por medio de la Interrupción 14H del BIOS, utilizando principalmente los registros AH, AL y DX para su configuración, como se muestra a continuación:

Función 00H de INT 14H: inicializa la comunicación con el puerto serial, con la velocidad de transmisión, la paridad, la longitud de palabra y el número de bits de paro deseados.

Es llamada con AH = 00H
AL = parámetros de inicialización *ver Nota 1*
DX = número de puerto de comunicación (0 = COM1, 1 = COM2, etc.)

Nota 1: los parámetros de inicialización son los siguientes:



Se presenta un programa de control con el Lenguaje de Programación Pascal, por medio del cual es posible controlar los movimientos del Brazo desde la PC, teniendo la posibilidad de generación automática de tareas, que en este caso es posible debido al control en tiempo real que se realiza ya que la interacción de la PC con el sistema es directo.

La utilización de Pascal, se fundamenta en el hecho de ser un lenguaje de alto nivel, con lo que es posible interactuar con el BIOS de la PC (para manejar las interrupciones), al igual que con sus periféricos; contando adicionalmente con la paquetería disponible en la PC, en caso de ser necesarias.

El programa en Pascal tiene la función de activar el puerto serial, y configurado para realizar la comunicación con la Sonda Robot utilizando el protocolo RS-232 para transmitir/recibir datos a 1200 Bauds también en tiempo Real. El programa permite mover el Brazo por medio del teclado, teniendo las opciones de grabar rutinas de control, utilizar rutinas previamente grabadas y paros de emergencia en caso de proximidad de impacto con algún objeto, así como, el control del desplazamiento de la sonda. Todas las señales de control son recibidas a través del UART de U1, donde son codificadas para luego ser enviada la secuencia de pasos correspondiente a cada motor vía los puertos del 82C55.

La configuración del puerto serial de la PC, se realiza por medio de la interrupción 14H del BIOS, utilizando principalmente los registros AH, AL y DX para su configuración, como se muestra a continuación:

Función 00H de INT 14H: inicializa la comunicación con el puerto serial, con la velocidad de transmisión, la paridad, la longitud de palabra y el número de bits de paro deseados.

Es llamada con AH = 00H
AL = parámetros de inicialización *ver Nota 1*
DX = número de puerto de comunicación (0 = COM1, 1 = COM2, etc.)

Nota 1: los parámetros de inicialización son los siguientes:

MSB							LSB
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Bits	7 6 5 Velocidad de Transmisión	4 3 Paridad	2 Bits de paro	1 0 Longitud de Palabra
	000 = 110 001 = 150 010 = 300 011 = 600 100 = 1200 101 = 2400 110 = 4800 111 = 9600	X0 = ninguna 01 = impar 11 = par	0 = 1 bit 1 = 2 bits	10 = 7 bits 11 = 8 bits

Para inicializar el puerto serial para velocidades de transmisión mayores a 9600 en una máquina PC del tipo PS/2, se utiliza INT 14H funciones 04h y 05H. (Para mayor información consultar referencia 5).

Función 01H de INT 14H: escribir un símbolo al puerto de comunicación.

Es llamada con AH = 01H
AL = símbolo
DX = número de puerto de comunicación (0 = COM1, 1 = COM2, etc)

Función 02H de INT 14H: leer un símbolo del puerto de comunicación.

Es llamada con AH = 02H
DX = número de puerto de comunicación (0 = COM1, 1 = COM2, etc)

Protocolo de Comunicación.

El puerto se configuró como sigue: 1200 Bauds, sin paridad, 1 bit de paro y 8 bits de datos, por lo que la palabra queda de la siguiente manera AL = 1000 0010 B = 82 H, mientras que el programa permite la utilización del puerto serial 1 o 2 según se desee.

4.3 Enlace al Puerto de Comunicación de una PC.

La conexión utilizada para realizar la interfase con el puerto serial de la PC, utilizando conectores DB9 y DB25, se muestra en la siguiente figura:

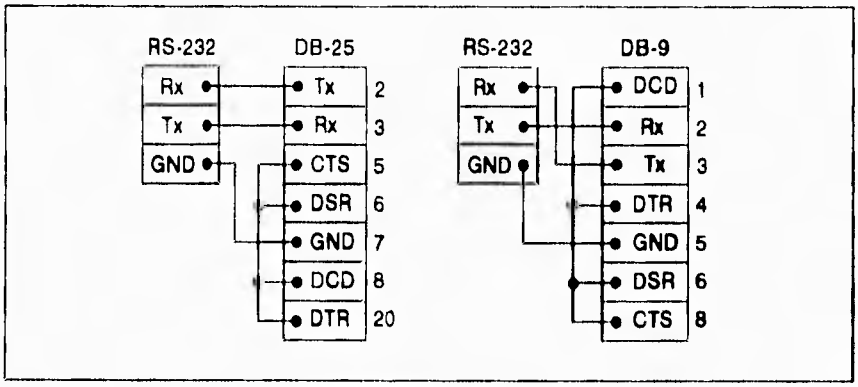


Figura 4.7. Utilización de los conectores DB-25 y DB-9.

4.4 Rutina de Control del Manipulador con la PC.

En la Figura 4.8 se muestra el diagrama de flujo del programa en PASCAL, la explicación del diagrama de flujo se presenta a continuación:

Inicialización.- Esta consta de los siguientes pasos:

1. El programa pregunta por el Puerto Serial al cual está conectada la Sonda Robot.

2. Se configura el Puerto Serial de la PC a 120 Bauds, con un bit de paro, 8 bits de datos y sin paridad como se explicó anteriormente.

3. Se limpia el puerto serial, operación que se efectúa al leer un dato del mismo, con lo que el registro AL tomará el valor de 00H.

Una vez realizado lo anterior el programa está preparado para recibir la orden a ejecutar, la cual es llamada en el programa LLAVE (introducido por medio de teclado).

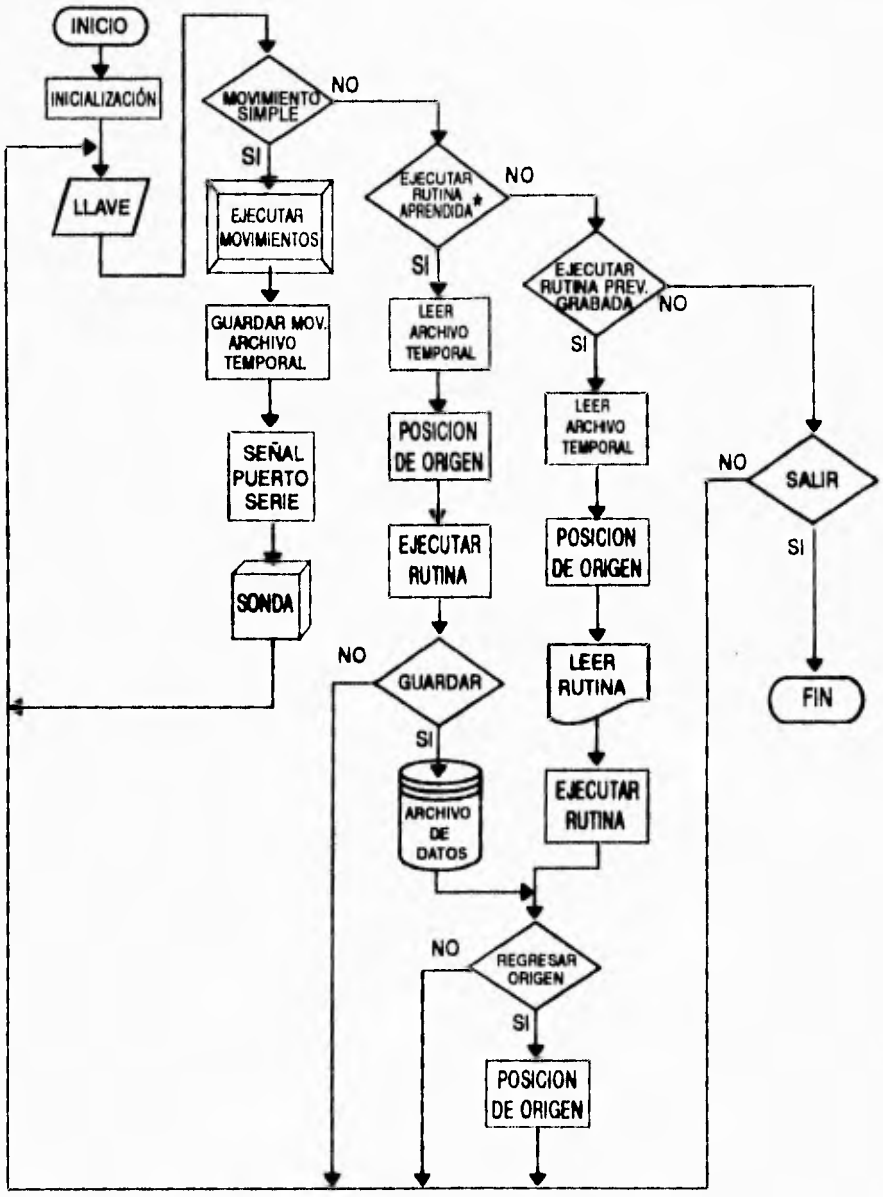


Figura 4.8. Algoritmo de Movimiento de un Manipulador para PC.

Se tienen tres posibles órdenes a ejecutar:

1. Movimiento simple: acciones disponibles para el Manipulador.

F1	Girar Base (Motor 3)
F2	Mover Codo (Motor 6)
F3	Mover Hombro (Motor 4)
F4	Mover Tenaza (Motor 5)
F5	Girar la Mano (Motor 1 y 2)
F6	Mover Mano (Motor 1 y 2)
F7	Mover Base y Codo
F8	Mover Hombro y Base
F9	Mover Hombro y Codo
F10	Mover Hombro, Base y Codo

2. Ejecutar Rutina Aprendida (SHIFT F1).

3. Ejecutar Rutina Previamente Grabada (SHIFT F2).

Una vez introducida la orden, el programa analiza la instrucción y determina cuál de las tres acciones se desea ejecutar.

Si la orden a ejecutar es un Movimiento Simple, se pasa al bloque siguiente llamado EJECUTAR MOVIMIENTO; donde se determina cuál de las diez acciones se desea realizar y la dirección deseada (acción que se efectúa por medio de las flechas del teclado). Una vez determinada la acción a efectuarse, se guarda el código correspondiente a la señal en un Archivo Temporal de la PC, para finalmente transmitirse a la Sonda (vía el Puerto Serie de la PC), donde se codifica para su posterior ejecución. Cada vez que sea ejecutada esta secuencia el archivo temporal se actualiza, es decir se crea una rutina, la cual será llamada RUTINA APRENDIDA.

En caso de recibir la orden de Ejecutar Rutina Aprendida, primero se regresará el Manipulador a la posición de origen (fijada por el operador), con lo que se lee el Archivo Temporal que fue creado para conocer la posición final del brazo. Ya en la posición de origen, se pasa al bloque de EJECUTA RUTINA donde son codificados cada uno de los códigos leídos del archivo temporal, para luego trasmitirse la señal correspondiente a la Sonda. Después se pregunta al operador si desea GUARDAR la rutina aprendida en disco duro (en caso afirmativo se pregunta por el nombre con el que se va a guardar). Enseguida se tiene la opción de regresar a la posición de origen si así se desea. Finalmente se limpia el Archivo Temporal para crear una nueva rutina.

Por último si la orden recibida es Ejecutar Rutina Previamente Grabada, se lee el archivo temporal para conocer la posición final del brazo y regresar a la posición de origen. A continuación se lee el Nombre de la Rutina a ejecutar y se leen los códigos uno a uno, los cuales son codificados en el bloque EJECUTAR RUTINA para ser enviadas las señales adecuadas a la Sonda. Una vez que se termina de ejecutar la rutina se limpia el archivo temporal y se pregunta al operador si desea regresar a la posición de origen.

4.5 Etapa de Potencia.

En cuanto a la etapa de potencia conectada al brazo mecánico (Diagrama Eléctrico 4), nos permite manejar una corriente máxima de 2.5 A por fase, con una Potencia Máxima de 65 Watts y un máximo de seis motores de pasos de cuatro fases (4 fases X 6 motores = 24 líneas = 3 Puertos).

Como se ve en el diagrama eléctrico de la etapa de potencia, se usa un TIP 120, el cual es un transistor tipo Darlington con una Beta mínima de 1000, con lo que podemos tener alta ganancia en corriente.

La operación de todos los transistores se efectúa en la región de corte o saturación. Con un uno (+5 V) en la base del transistor opera en la región de saturación, con lo que la fase correspondiente del motor se polariza. Con un cero en la base del transistor trabaja en la región de corte por lo que no está polarizada la fase del motor. El diodo D2 es utilizado para protección tanto del transistor como del motor, puesto que al apagar la fase de motor se genera un pico de alto voltaje el cual es disipado por medio del diodo en forma de calor.

El circuito cuenta con un diodo de señal (D1) para evitar el paso de corrientes inversas en caso de que se rompa la juntura del transistor que va de la base al colector; asimismo cuenta con una resistencia R2 conectada a tierra, para descargar el circuito, a fin de eliminar la capacitancia que se forma entre la base y el emisor del transistor.

EL Sistema de tracción utiliza motores de DC operando en lazo abierto. El control se realiza por medio del puerto C del 82C55 con los 4 bits más significativos. Los dos bits más altos (C_7 y C_6) se utilizan para la tracción y los 2 más bajos (C_5 y C_4) para la dirección.

Este sistema se basa en la técnica utilizada en los "Chopers" clase E, para el cual se están utilizando dos circuitos, uno para la tracción y el otro para la dirección. El "Choper" clase E se

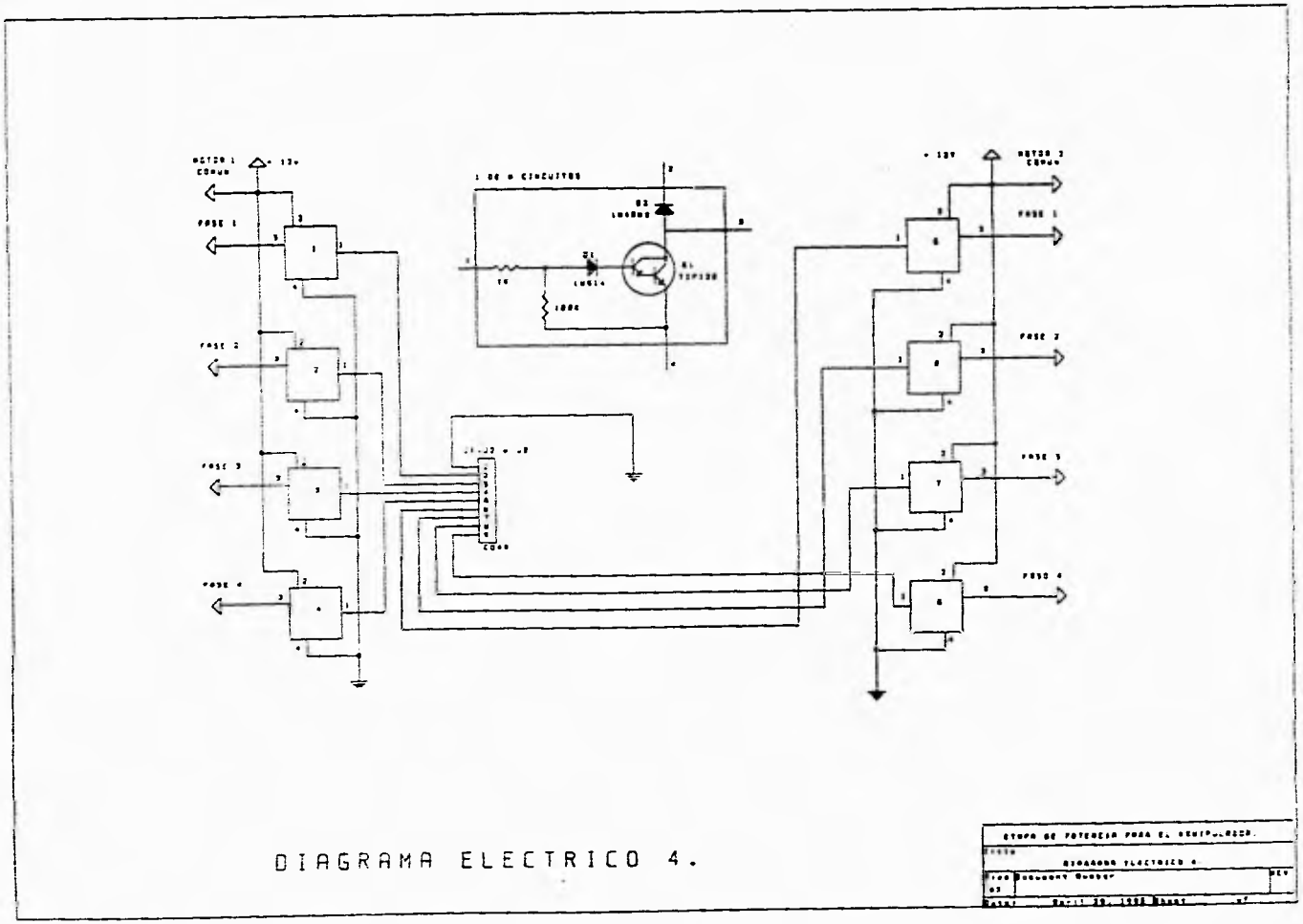


DIAGRAMA ELECTRICO 4.

muestra en el Diagrama Eléctrico 5. Para la conmutación se decidió utilizar transistores de alta potencia, debido a que las pruebas realizadas con SCR's y GTO's presentaron mayores pérdidas que los transistores, así como muchos problemas para lograr la conmutación, debido a la necesidad de realizar conmutación forzada para apagarlos, lo que resultaba en un circuito mucho más complejo si se compara con el que se utilizó.

La elección de utilizar un "Choper" se debe a sus numerosas ventajas para el control de motores de DC, como son su alta eficiencia, flexibilidad en control, peso ligero, tamaño reducido, respuesta rápida, y baja regeneración para velocidades muy bajas. Además de que resulta ideal para vehículos que operan con baterías.

Debido a la elevada frecuencia de la ondulación del voltaje de salida en un Choper, la ondulación de corriente de armadura en el motor es menor lo que reduce las pérdidas y los efectos negativos en la potencia de salida del motor. Con la utilización de transistores de potencia, la frecuencia de operación puede ser mayor a 2.5 KHz.

La operación del circuito (Diagrama Eléctrico 5) es como sigue: si se quiere avanzar se polariza con un uno lógico el bit C_7, con lo que conducen S1 y S2. Si se quiere retroceder se polariza C_6, conduciendo S3 y S4. Los diodos son utilizados para protección contra picos de alto voltaje, el cual se genera al desactivar alguno de los transistores.

Para el caso de la dirección, opera de la misma forma energizando C_5 se gira en sentido horario y energizando C_4 se gira en sentido antihorario.

4.6 Cálculo del Tamaño del Disipador.

El uso de un disipador en un dispositivo semiconductor puede tener un tremendo impacto en la temperatura de operación del componente. Para algunos dispositivos, un buen disipador puede hacer la diferencia entre un proyecto exitoso y un fracaso. El propósito de un disipador es mover la mayor cantidad posible de calor fuera de la juntura del dispositivo, y esto se logra con la selección del disipador, del arreglo para el montaje y de los materiales a emplear en el montaje.

El calor es generado por un dispositivo semiconductor cuando éste disipa energía. La energía disipada por un transistor es el voltaje de caída entre el colector y el emisor multiplicada por la corriente de colector (en el caso de utilizar señales cuadradas se multiplicará también por el ciclo de trabajo).

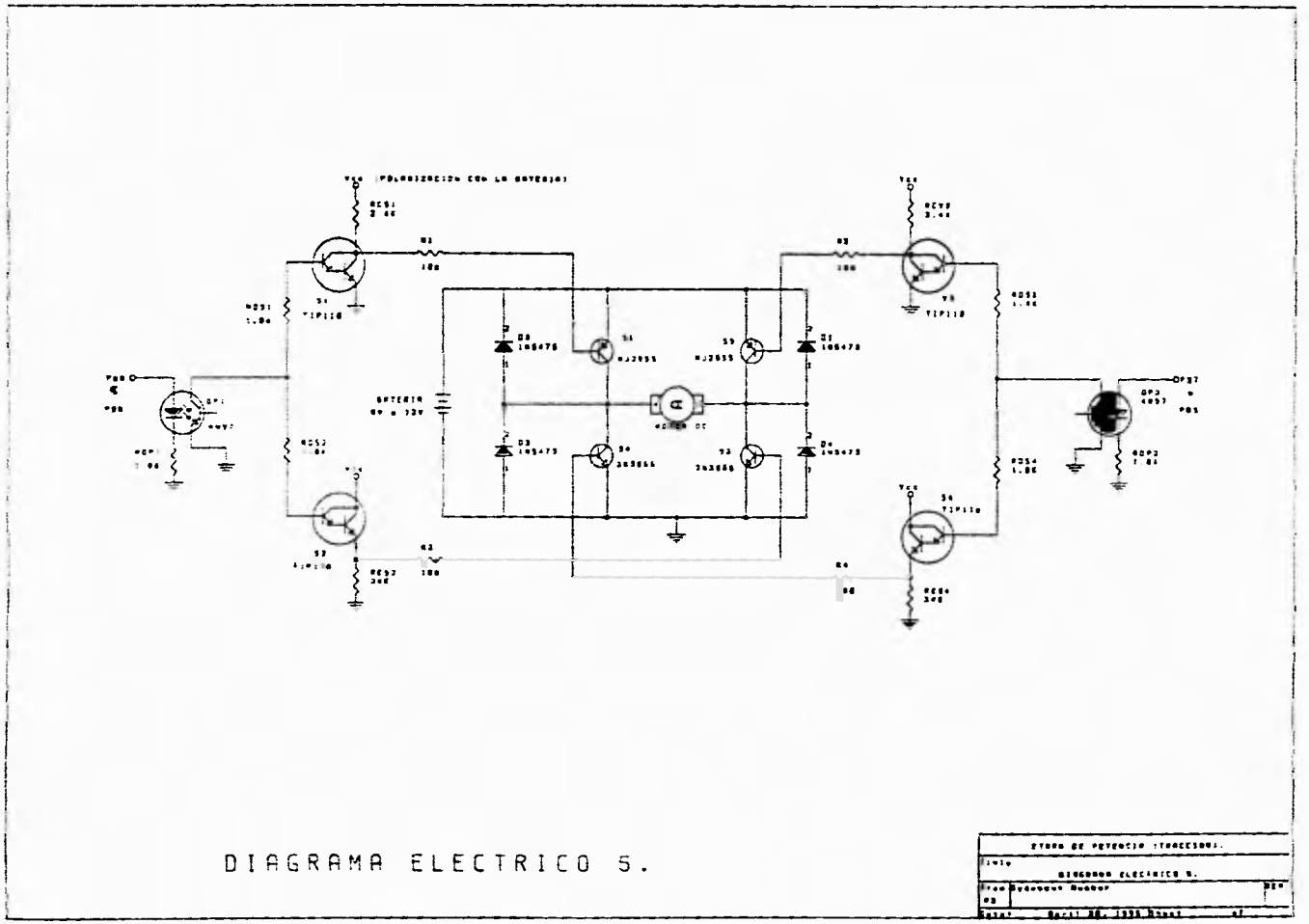


DIAGRAMA ELECTRICO 5.

Para el cálculo del disipador es necesario conocer el concepto de resistencia térmica, que se define generalmente como la diferencia en temperatura entre dos puntos, dividida por la potencia que se disipa entre esos dos puntos. La resistencia térmica es simbolizada por la letra Griega theta y se mide en grados Celsius por watt (°C/W). Como regla, la resistencia térmica debe ser tan pequeña como sea posible entre la juntura del semiconductor que disipa energía y el aire del ambiente. Una valor de resistencia térmica bajo entre las junturas asegura una temperatura de juntura baja.

El uso de un disipador externo típicamente involucra tres resistencias térmicas: 1) entre la juntura del semiconductor y el casco del dispositivo, 2) entre el casco y el disipador fijado al dispositivo, y 3) entre el disipador y el aire del ambiente. Una resistencia térmica adicional puede ser encontrada si un aislante eléctrico es incluido entre el casco y el disipador.

Para componentes que no usan un disipador externo, dos resistencias térmicas deben ser consideradas: 1) entre la juntura del semiconductor y el casco, y 2) entre el casco y el aire del ambiente. Las siguientes fórmulas resumen el cálculo de la resistencia térmica total para los diferentes arreglos:

$\theta_{JA}=\theta_{JC}+\theta_{CS}+\theta_{SA}$ A) Resistencia térmica total solamente con disipador

$\theta_{JA}=\theta_{JC}+\theta_{CS}+\theta_{INS}+\theta_{SA}$ B) Resistencia térmica total con disipador y aislante eléctrico

$\theta_{JA}=\theta_{JC}+\theta_{CA}$ C) Resistencia térmica total sin disipador

θ_{JA} = Resistencia térmica de la juntura al ambiente
θ_{CS} = Resistencia térmica del casco al disipador
θ_{JC} = Resistencia térmica de la juntura al casco
θ_{SA} = Resistencia térmica del disipador al ambiente
θ_{INS} = Resistencia térmica del aislante eléctrico
θ_{CA} = Resistencia térmica del casco al ambiente

Todos los valores de θ están en °C/W

La resistencia térmica formada por la barrera del casco y el disipador (θ_{CS}) está en función de muchos factores: 1) al área de contacto de la sección transversal entre el casco y el disipador, 2) el terminado y lo parejo de la superficie de contacto, 3) la fuerza de contacto (o presión) aplicada entre las superficies de contacto, y 4) la resistencia térmica de cualquier aislante eléctrico necesitado entre el casco y el disipador.

Para el cálculo del disipador es necesario conocer el concepto de resistencia térmica, que se define generalmente como la diferencia en temperatura entre dos puntos, dividida por la potencia que se disipa entre esos dos puntos. La resistencia térmica es simbolizada por la letra Griega theta y se mide en grados Celsius por watt (°C/W). Como regla, la resistencia térmica debe ser tan pequeña como sea posible entre la juntura del semiconductor que disipa energía y el aire del ambiente. Una valor de resistencia térmica bajo entre las junturas asegura una temperatura de juntura baja.

El uso de un disipador externo típicamente involucra tres resistencias térmicas: 1) entre la juntura del semiconductor y el casco del dispositivo, 2) entre el casco y el disipador fijado al dispositivo, y 3) entre el disipador y el aire del ambiente. Una resistencia térmica adicional puede ser encontrada si un aislante eléctrico es incluido entre el casco y el disipador.

Para componentes que no usan un disipador externo, dos resistencias térmicas deben ser consideradas: 1) entre la juntura del semiconductor y el casco, y 2) entre el casco y el aire del ambiente. Las siguientes fórmulas resumen el cálculo de la resistencia térmica total para los diferentes arreglos:

$\theta_{JA}=\theta_{JC}+\theta_{CS}+\theta_{SA}$ A) Resistencia térmica total solamente con disipador

$\theta_{JA}=\theta_{JC}+\theta_{CS}+\theta_{INS}+\theta_{SA}$ B) Resistencia térmica total con disipador y aislante eléctrico

$\theta_{JA}=\theta_{JC}+\theta_{CA}$ C) Resistencia térmica total sin disipador

θ_{JA} = Resistencia térmica de la juntura al ambiente
θ_{CS} = Resistencia térmica del casco al disipador
θ_{JC} = Resistencia térmica de la juntura al casco
θ_{SA} = Resistencia térmica del disipador al ambiente
θ_{INS} = Resistencia térmica del aislante eléctrico
θ_{CA} = Resistencia térmica del casco al ambiente

Todos los valores de θ están en °C/W

La resistencia térmica formada por la barrera del casco y el disipador (θ_{CS}) está en función de muchos factores: 1) al área de contacto de la sección transversal entre el casco y el disipador, 2) el terminado y lo parejo de la superficie de contacto, 3) la fuerza de contacto (o presión) aplicada entre las superficies de contacto, y 4) la resistencia térmica de cualquier aislante eléctrico necesitado entre el casco y el disipador.

Es a menudo deseable estimar la temperatura de la juntura de un semiconductor para determinar si trabajará dentro de rangos seguros de operación. Está técnica puede ser manejada para estimar el desempeño del disipador. La máxima temperatura de juntura (T_j) de un dispositivo semiconductor es normalmente especificado en la hoja de datos del fabricante. La temperatura de la juntura puede ser calculada de la relación mostrada a continuación:

$$T_j = (P_d)(\theta_{JA}) + T_a \quad \text{.....Ecuación 2.}$$

T_j = temperatura de la juntura (°C)
P_d = potencia disipada (watts)
θ_{JA} = resistencia térmica total (°C/W)
T_a = temperatura ambiente (°C)

A continuación se procede al cálculo de la temperatura de juntura para un transistor TIP 120, utilizado para manejar cada una de las fases de los motores de pasos. Este tipo de transistor presenta un casco TO-220 cuya temperatura de juntura (T_j) máxima utilizada militarmente es del orden de los 105 °C. El transistor es utilizado en la región de corte y en la de saturación, con lo que el Vce = 0.2 V, con una corriente de colector Ic = 0.8 A. El ciclo de trabajo es del 95%, la frecuencia máxima de conmutación es de 10 Hz y la temperatura ambiente T_a = 25 °C.

Debido a que el transistor se utiliza en corte y saturación la energía disipada en forma de calor es mínima y como la frecuencia de trabajo es baja, las pérdidas en el motor son despreciables; lo que da la posibilidad de no requerir un disipador, como se observa a continuación:

Del manual de datos de Motorola (referencia 11)

θ_{JA} = = 62.5 °C/W (Para el tipo TO-220 sin disipador)

De la ecuación 2:

T_j = [(0.2 V)(0.8 A)(0.95)][62.5 °C/W] + 25 °C

T_j = [0.152 W][63 °C/W] + 25 °C = 9.5 °C + 25 °C

T_j = 34.5 °C como el resultado está muy por debajo de la especificación máxima, se puede garantizar que el transistor operará adecuadamente sin disipador.

El siguiente cálculo de la temperatura de juntura es para el caso de los transistores 2N3055 y para el MJ2955 empleados en los "Chopers" para la tracción y la dirección. El tipo de casco es el TO-3 cuya temperatura de juntura máxima (T_j) utilizada militarmente es del orden de los 105 °C. Los transistores son utilizados en la región de corte y en la de saturación por lo que Vce = 0.2 V, con una corriente de colector de 2.0 A. El ciclo de trabajo es del 98%, con una frecuencia máxima de conmutación de 2 Hz y una temperatura ambiente T_a = 25 °C.

Debido a que el transistor se utiliza en corte y saturación la energía disipada en forma de calor es mínima y como la frecuencia de trabajo es baja, las pérdidas en el motor son despreciables (del orden de los 48 microwatts); lo que da nuevamente la posibilidad de no requerir disipador, como se demuestra a continuación:

Del manual de datos de Motorola (referencia 11) y de la Tabla 6 de la referencia 12

$\theta_{JA}=\theta_{JC}+\theta_{CA}=$ 1.52 °C/W + 30.0 °C/W (Para el tipo TO-3 sin disipador)

De la ecuación 2:

T_j = [(0.2 V)(2 A)(0.98)][31.52 °C/W] + 25 °C

T_j = [0.392 W][31.52 °C/W] + 25 °C = 12.35 °C + 25 °C

T_j= 37.35 °C como el resultado está muy por debajo de la especificación máxima, se puede garantizar que el transistor operará adecuadamente sin disipador.

Tipo de chasis	θ_{JC}[°C/W] (típica)	θ_{CA}[°C/W] (típica)	θ_{CS}[°C/W]	θ_{INS}[°C/W] (Mica)
TO-3 (TO-204)	1.5	30.0	0.1 a 0.7	1.6
TO-220	3.0	60.0	0.9 a 1.3 (atornillado)	5.2

4.7 Sistema de Adquisición de Datos

La adquisición de datos se realiza por medio del circuito ADC0817 (U6) el cual es un convertidor analógico-digital de 8 bits y 16 canales multiplexados, presentando un error total no ajustable de tan sólo $\pm 1 LSB$ en la conversión. Utiliza la técnica de aproximación sucesiva en la conversión, la cual se efectúa en tan sólo 100 microsegundos. La siguiente figura muestra en bloques la etapa completa del sistema de adquisición de datos y el Diagrama Eléctrico 6 la interconexión de los componentes utilizados en esta etapa.

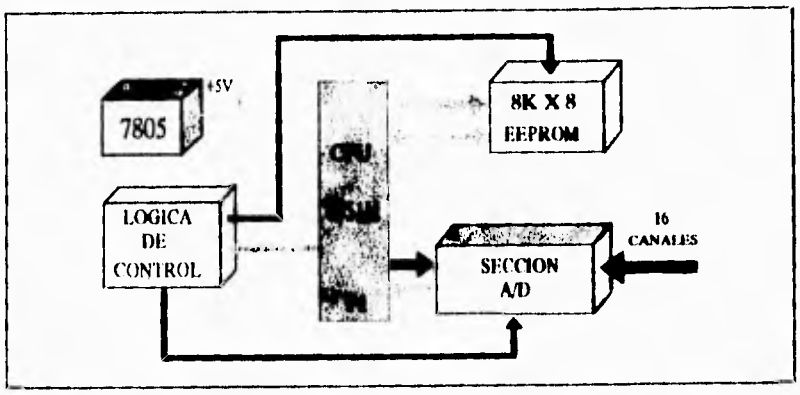


Figura 4.9. Diagrama de Bloques de la Etapa de Adquisición de Datos.

La Figura 4.10 muestra los diagramas de tiempos para la activación del Convertidor; como se observa, es necesario realizar una escritura indicando el canal a ser leído con el acumulador. Al realizar esta operación la señal WR (negada) efectúa una transición de alto a bajo. Esta señal es conectada directamente a la entrada de un inversor (U7B), por lo que la salida es un uno lógico que va directamente a la entrada de una AND (U8A), que en conjunto con la señal de dirección del ADC (E000H como se indica en la Figura 4.2a) producen un uno lógico a la salida de la compuerta, dando como resultado la activación de las terminales ALE y START del ADC0817, por lo que se lee el canal seleccionado y se inicia la conversión.

Durante este proceso la señal EOC del ADC0817 (señal de salida para indicar la finalización de la conversión), realiza una transición de alto a bajo mientras se efectúa la conversión internamente. Un aspecto que se debe hacer notar es la configuración de la Interrupción Externa 1 del 8751H (INT 1), pues es esta interrupción como ya se mencionó anteriormente, la que atiende la petición de interrupción realizada por el ADC, petición que se hace por medio de la señal EOC, la cual al realizar la transición de alto a bajo activaría a INT 1 que fue programada para activarse por flanco de bajada.

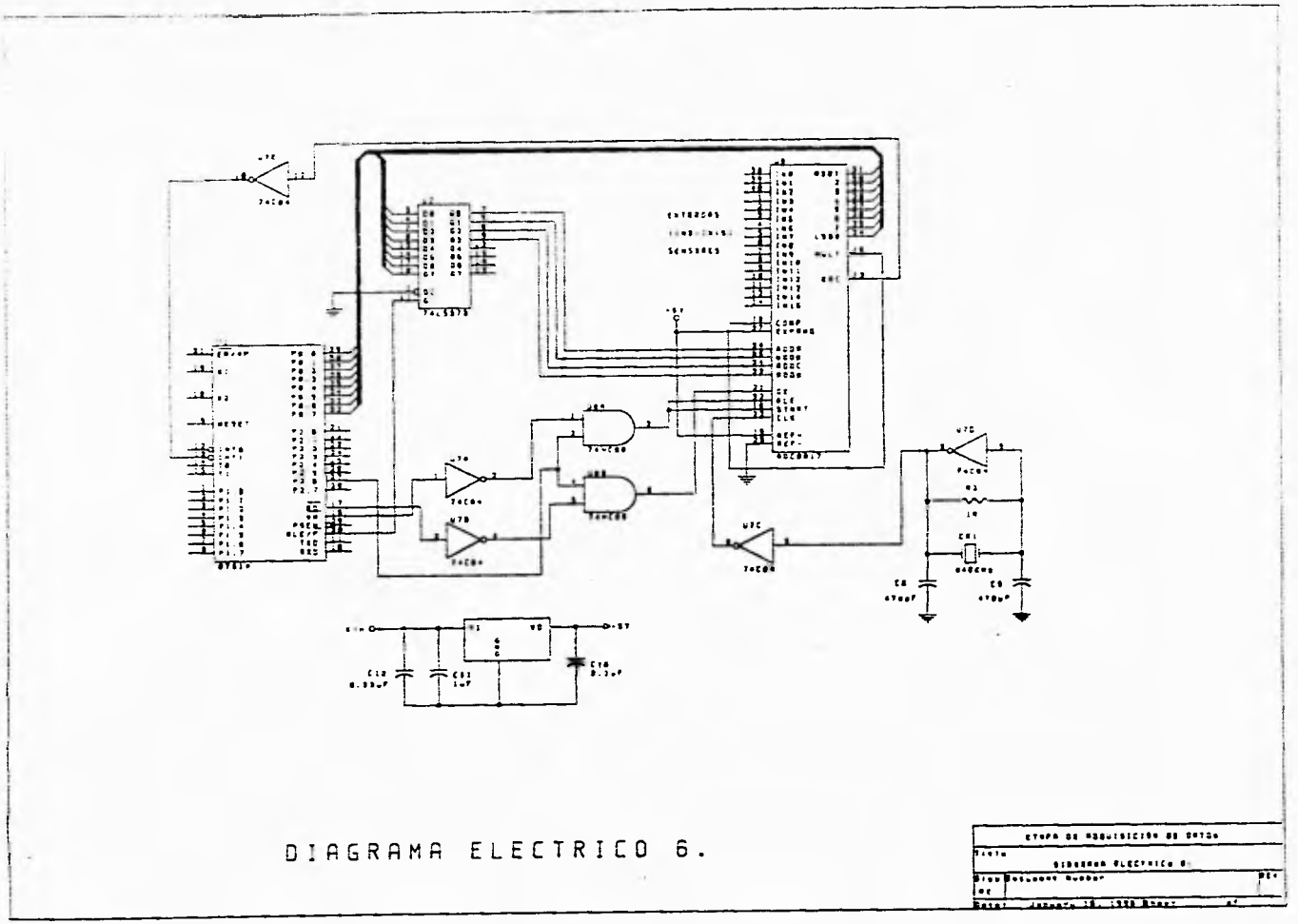


DIAGRAMA ELECTRICO 6.

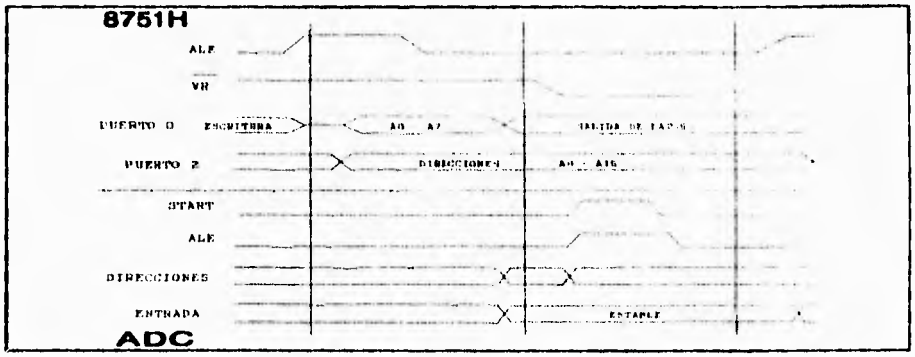


Figura 4.10. Diagrama de Tiempos del ADC0817 (Activación).

Analizando los diagramas de tiempos del 8751H y del ADC0817, se observó que si EOC se conectaba a un inversor (U7E como se muestra en el Diagrama Eléctrico 6), la señal de salida correspondiente realizará una transición de bajo a alto evitando la activación de INT 1. Mientras que al terminarse la conversión (momento en el que sí debe activarse INT 1 pues el dato ya está listo para ser leído), la señal EOC realiza otra transición, pero ahora de bajo a alto; señal que al entrar al inversor produce una señal de salida con flanco de bajada, que es el momento en que se activa INT 1. Una vez atendida esta interrupción, se llama la rutina de servicio correspondiente en U1, la cual realiza un acceso de lectura al Bus de datos del dispositivo externo (ADC0817), con lo que la señal RD (negada) realiza una transición de alto a bajo; señal que al entrar al inversor (U7B) produce un uno lógico a la salida de éste. La salida de U7B está conectada directamente a la compuerta AND (U8B) que junto con la señal de dirección del ADC (E000H) producen un uno lógico a la salida de la compuerta, señal que activa la terminal OE del ADC; con lo que el bus de datos del convertidor está listo para ser accesado.

Al ser tomada la lectura del bus de datos del ADC, ésta se guarda en el Acumulador para poder ser manipulada. La siguiente figura muestra los diagramas de tiempos para realizar la lectura de datos en el ADC0817.

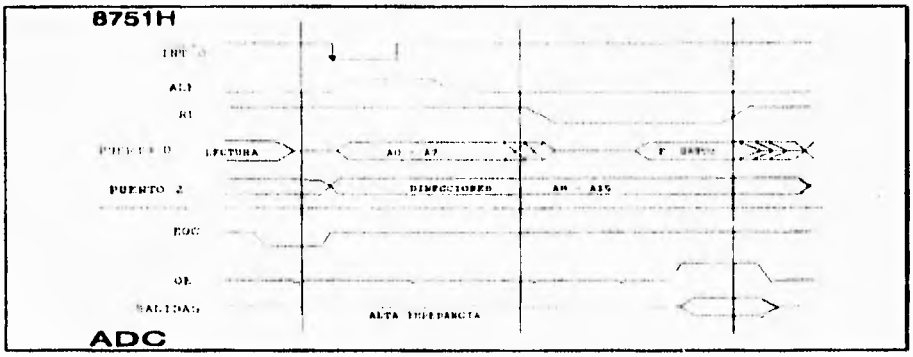


Figura 4.11. Diagramas de Tiempos del ADC0817 (Lectura).

A continuación se mencionan los pasos a seguir para leer datos de alguno de los 16 canales del ADC0817, así como, la rutina en lenguaje ensamblador.

Primero es necesario conocer la dirección de memoria en que está mapeado el ADC. Observando las Figuras 4.2a y 4.2b, una de las direcciones corresponde a la localidad de memoria E000H. Conocida la dirección tan sólo es necesario determinar el canal a leer, como muestra la rutina siguiente:

MOV DPTR,#E003H	Se carga la dirección del canal del ADC0817 en el registro DPTR de 16 Bits. En éste caso se eligió el Canal 4 (E003H).
MOVX @DPTR,A	Se realiza una operación de escritura a un dispositivo externo (MOVX), con lo que es activado el canal 1 del ADC

La activación de los canales del ADC en el programa de control se realiza con la rutina llamada ACTIVACIÓN DE CANALES DEL ADC, indicándose el código con el cual se activa cada canal analógico.

Ya activado el canal 1 del ADC, se activa la interrupción externa uno (IE1) del microcontrolador una vez que ha concluido el tiempo de conversión (procedimiento explicado anteriormente), llamándose la rutina de servicio correspondiente (EXT1).

EXT1:	MOV DPTR,#E000H	Se carga la dirección del ADC0817
	MOVX A,@DPTR	Se lee el bus de datos del ADC0817, que es una lectura de 8 bits, la cual se almacena en el acumulador para manipularla

La rutina de servicio se identifica en el programa de control como RUTINA DE SERVICIO PARA INTERRUPCIÓN EXTERNA 1.

4.8 Almacenamiento de Datos.

El sistema cuenta con una memoria EEPROM 2864 (U3) como se muestra en el Diagrama Eléctrico 7, de 8Kb x 8 (8191 localidades de 8 bits) que por ser eléctricamente grabable y borrable, permite el almacenamiento de los datos adquiridos por medio del Convertidor U6.

Entre sus características se menciona que es una memoria de alta velocidad (aproximadamente 250 ns máximo para lectura y 30 ms como máximo para escritura), presenta un circuito para asegurar la integridad de los datos durante el encendido o el apagado, 10 años de retención de datos por cada escritura y un mínimo de 10,000 ciclos de escritura/lectura por byte.

La siguiente figura muestra el diagrama de tiempos del 8751H y de la EEPROM. Su habilitación se realiza con P2.5 en bajo, debido a que el habilitador de la EEPROM (CE) es negado.

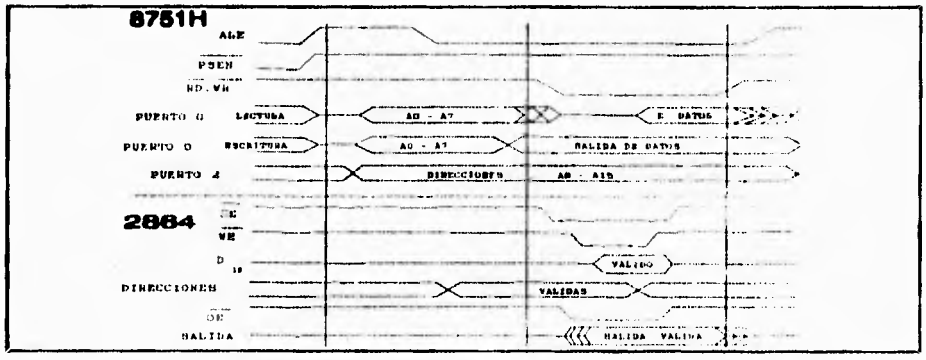


Figura 4.12. Diagramas de Tiempos del 8751H y de la EEPROM.

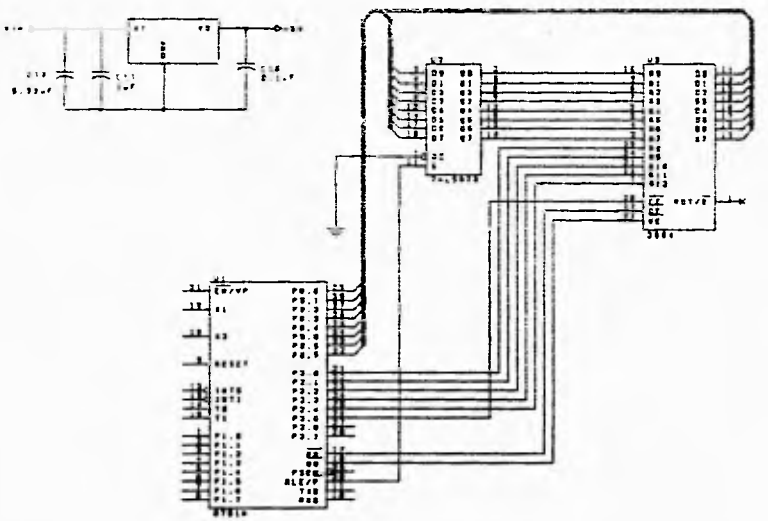


DIAGRAMA ELECTRICO 7.

Las siguientes rutinas en ensamblador muestran la forma de realizar la escritura y la lectura de datos.

Rutina de Escritura:

MOV DPTR,#8000H	Se carga la localidad de memoria a la cual se va a escribir. En este caso se selecciona la 8000H que corresponde a la primera localidad de la EEPROM, según se observa en las Figuras 4.2a y 4.2b
MOVX @DPTR,A	Se guarda el dato que contiene el Acumulador a la localidad seleccionada anteriormente

En la Rutina de Servicio de la Interrupción Externa 1 se localiza la Rutina de Escritura, indicándose con etiquetas el momento en que se efectúan las operaciones necesarias para esta función.

Rutina de Lectura:

MOV DPTR,#9000H	Se carga la localidad de memoria que quiere ser accesada. Se selecciona la localidad #9000H
MOVX A,@DPTR	El contenido de la localidad de memoria se carga en el Acumulador para poder manipularla

Los datos que serán almacenados en la memoria son el tiempo, la distancia, orientación, radiación beta, radiación gama y temperatura, es decir seis datos; por lo que es posible almacenar un total de 1365 conjuntos (cada uno de seis datos).

La rutina mostrada a continuación es empleada para trasmitir el contenido de la memoria a una computadora:

MOV A,#A0H	Carga el Acumulador con A0H
MOV DPTR,#8000H	Dirección de la primera localidad de la EEPROM que será accesada
T: PUSH ACC	Salva el contenido del Acumulador en RAM
MOVX A,@DPTR	Transfiere el contenido de la dirección al Acumulador (lectura)

MOV SBUF,A	Transfiere el contenido del Acumulador al Almacenador Serial de Datos
JNB TI,$	Instrucción utilizada para determinar la finalización de la transmisión
CLR TI	Limpia la bandera que se activa durante la transmisión
INC DPTR	Direccionamos la siguiente localidad de la EEPROM
POP ACC	Carga el Acumulador con A0H
CJNE A,DPH,T	Compara el Acumulador con el Byte Alto de DPTR, en caso de que sean diferentes salta a la instrucción con etiqueta T (PUSH ACC). Si son iguales termina la transmisión de los datos de la EEPROM y ejecuta la siguiente instrucción

La Rutina anterior se identifica en el programa de control como **RUTINA PARA TRANSMITIR DATOS DE LA EEPROM.**

CAPITULO 5

OPERACIÓN DEL EQUIPO

En éste capítulo se describen los pasos que se deben seguir para poder armar el prototipo, así como las pruebas para verificar el correcto funcionamiento del mismo.

5. OPERACIÓN DEL EQUIPO.

El equipo completo consta de cuatro tarjetas de circuito impreso, dos baterías recargables (6 y 12 volts), seis interfases de cable plano para conexión a las tarjetas, una interfase para conexión con el puerto serial de una computadora, programa de control para el microcontrolador realizado en ensamblador y el programa de control para la PC realizado en Pascal.

Los requerimientos para poder operar el equipo son los siguientes:

- Una Computadora Personal con microprocesador 286 o superior, con sistema operativo 3.0 o posterior, monitor a color (recomendable) y con un puerto serial disponible.

- Un ensamblador modelo CYS-8051 versión 3.02 compatible con IBM PC (Diseñado por Cybernetic Micro Systems).

- Un editor de texto compatible con IBM PC.

- Un grabador de memorias y microprocesadores modelo MOD-MUP (C) diseñado por Modular Circuit Technology. En caso de contar con algún otro modelo éste podrá utilizarse si cuenta con el software para el Microcontrolador 8751H de Intel.

A continuación se mencionan los pasos que se deben seguir antes de que el equipo esté listo para operar:

1. **Editar en la PC el programa monitor** que se presenta en el Apéndice A.

2. **Ensamblar el programa monitor;** para lo cual se debe correr el paquete ensamblador CYS-8051 (CYS8051.EXE). Una vez realizada esta acción el paquete presentará el siguiente menú de acciones:

The Cybernetic Micro Systems
CYS-8051 Software Package
Version 3.02
Serial Number 001307.8051.....
for the IBM Personal Computer

(C) Copyright Cybernetic Micro Systems, Inc 1983, 1984, 1985, 1986

High Level Languaje Commands:

ASM51 [file]	P8751 file	P8744 file	
P27xx file	COMP [file]	READ [Kbytes]	DISP [start,num]

INITIALIZE	CLEAR	LIST	EXIT
RUN [file]	GOTO label	STOP	CONT
DIR [spec]	TYPE file	DOS cmd	HELP
PRINT arglist	EQU	=	WARBLE
DELAY [secs]	CLS	HLSEARCH flag	BEEP
OFFSET pc[,loc]	PROM type	SECURE	SAVE file[,start,num]
LOAD file	EDBYTE pc	PVOLT value	PSETUP num,value

>

Como se observa se encuentra el comando ASM51, que al teclearlo junto con la ruta completa donde fue editado el programa monitor el paquete procederá a ensamblar el programa, verificando que no exista ningún error de código en el programa. Una vez que no se detecte algún error, el programa será ensamblado con código hexadecimal.

> ASM51 C:\TURBO\MONITOR.PAS
00 Errors
> Exit (comando para salir del programa)

3. **Cambiar el programa monitor de código hexadecimal a código binario.** Entrar al directorio del paquete MOD-MUP y correr el programa HEXOBJ, donde se tecleará la ruta del programa monitor en hexadecimal, seleccionándose el formato de INTEL para la transformación a código binario.

4. **Borrar la EPROM del microcontrolador 8751H** con un borrador de memorias o con una lámpara que emita luz ultravioleta.

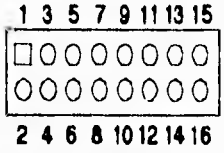
5. **Grabar el programa monitor con código binario en el 8751H.** Se ejecuta el paquete MOD-MUP (C) (51P02.EXE), apareciendo en el monitor un menú de acciones. Se selecciona el tipo de circuito que será grabado al presionar la letra T como se indica en el menú de acciones (Type), con lo que aparecerá un submenú con lo diferentes tipos de circuitos, que en este caso corresponderá al número 3 (8751H). A continuación se carga el programa monitor con código objeto, acción que se lleva acabo al seleccionar el número 2 del menú de acciones, donde se preguntará la ruta de ubicación del programa y la dirección de inicio (0000H). Se coloca el circuito integrado en la base disponible. Finalmente se procede a la programación del 8751H al teclear la letra P (PROGRAM), donde se indicará la dirección de inicio de la memoria (0000H), la última dirección de la memoria (0FFFH) y la dirección de inicio del bus de datos (0000H). Al finalizar la programación el paquete escribirá una señal de aviso indicándose el resultado de la operación.

6. **Colocar el microcontrolador en la base correspondiente (U1).** (Tarjeta principal cque contiene el microcontrolador).

7. Realizar las conexiones de todas las interfases de cable plano y de la interfase para conexión con el puerto serial de la PC.

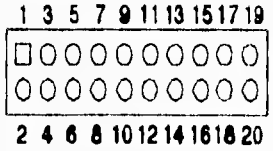
La tarjeta principal cuenta con seis conectores indicados con las etiquetas J1, J2, J3, J4, J5 y J6.

- El conector J1 está disponible para conectar las señales adicionales (8 bits, niveles TTL) utilizadas por el microcontrolador (U1), para determinar la orientación de la Sonda Robot.



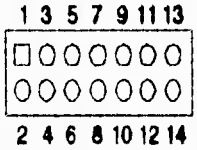
Terminales 1,3,5 y 11 no están conectadas
Terminal 2: P1.0
Terminal 4: P1.1
Terminal 6: P1.2
Terminales 7 y 9: Tierra
Terminal 8: P1.3
Terminal 10: P1.4
Terminal 12: P1.5
Terminales 13 y 15: Voltaje (+ 5V)
Terminal 14: P1.6
Terminal 16; P1.7

- Conector J2: utilizado para las entradas del Convertidor Analógico-Digital (ADC0817).



Terminal 1: Canal 3
Terminal 2: Canal 4
Terminal 3: Canal 2
Terminal 4: Canal 5
Terminal 5: Canal 1
Terminal 6: Canal 6
Terminales 7 y 9: Tierra
Terminal 8: Canal 7
Terminal 10: Canal 8
Terminal 11: Canal 16
Terminal 12: Canal 9
Terminal 13: Canal 15
Terminal 14: Canal 10
Terminal 15: Canal 14
Terminal 16: Canal 11
Terminales 17 y 19: Voltaje (+5V)
Terminal 18: Canal 12
Terminal 20: Canal 13

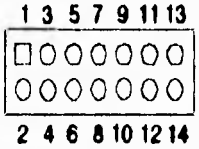
- El conector J3 se conecta a la Etapa de Potencia 1 (Apéndice D) en el lugar donde se encuentra la etiqueta J3 mediante la interfase de cable plano de 14 hilos.



Terminal 1: Salida PA4 del 82C55
Terminal 2: Salida PA3 del 82C55
Terminal 3: Salida PA5 del 82C55
Terminal 4: Salida PA2 del 82C55
Terminal 5: Salida PA6 del 82C55
Terminal 6: Salida PA1 del 82C55
Terminal 7: Salida PA7 del 82C55
Terminal 8: Salida PA0 del 82C55
Terminales 9 y 11: Tierra
Terminal 10: Salida PC7 del 82C55

Terminal 1: Canal 3
Terminal 2: Canal 4
Terminal 3: Canal 2
Terminal 4: Canal 5
Terminal 5: Canal 1
Terminal 6: Canal 6
Terminales 7 y 9: Tierra
Terminal 8: Canal 7
Terminal 10: Canal 8
Terminal 11: Canal 16
Terminal 12: Canal 9
Terminal 13: Canal 15
Terminal 14: Canal 10
Terminal 15: Canal 14
Terminal 16: Canal 11
Terminales 17 y 19: Voltaje (+5V)
Terminal 18: Canal 12
Terminal 20: Canal 13

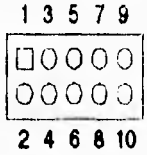
- El conector J3 se conecta a la Etapa de Potencia 1 (Apéndice D) en el lugar donde se encuentra la etiqueta J3 mediante la interfase de cable plano de 14 hilos.



Terminal 1: Salida PA4 del 82C55
Terminal 2: Salida PA3 del 82C55
Terminal 3: Salida PA5 del 82C55
Terminal 4: Salida PA2 del 82C55
Terminal 5: Salida PA6 del 82C55
Terminal 6: Salida PA1 del 82C55
Terminal 7: Salida PA7 del 82C55
Terminal 8: Salida PA0 del 82C55
Terminales 9 y 11: Tierra
Terminal 10: Salida PC7 del 82C55

Terminal 12: Salida PC6 del 82C55
Terminal 13: Salida PC4 del 82C55
Terminal 14: Salida PC5 del 82C55

- Los conectores J4, J5 y J6 se conectan a las etiquetas correspondientes en la Etapa de Potencia 2 (Apéndice E). La conexión se realiza por medio de tres interfases de cable plano de 10 hilos cada una.



Conector J4: Terminales 1, 3, 5, 7, 9 y 10: Tierra
Terminal 2: Salida PC0 del 82C55
Terminal 4: Salida PC1 del 82C55
Terminal 6: Salida PC2 del 82C55
Terminal 8: Salida PC3 del 82C55

Conector J5: Terminales 1, 3, 5, 7, 9 y 10: Tierra
Terminal 2: Salida PB0 del 82C55
Terminal 4: Salida PB1 del 82C55
Terminal 6: Salida PB2 del 82C55
Terminal 8: Salida PB3 del 82C55

Conector J6: Terminales 1, 3, 5, 7, 9 y 10: Tierra
Terminal 2: Salida PB4 del 82C55
Terminal 4: Salida PB5 del 82C55
Terminal 6: Salida PB6 del 82C55
Terminal 8: Salida PB7 del 82C55

En la misma tarjeta principal (8751H) se cuenta con diversos conectores disponibles para las siguientes funciones:

Nota: los conectores J9, J10, J11, J12 y J13 para este caso no son conectados (sin uso) sólo se dejan para posibles aplicaciones de alguna señal que un futuro sea necesaria.

Conector J9: Entrada disponible al inversor 74HC04 (U7F)

Terminal 1: Entrada a U7F
Terminal 2: Vcc (+5V)
Terminal 3: Tierra

Conector J10: Salida del inversor U7F

Terminal 1: Salida de U7F
Terminal 2: Vcc (+5V)
Terminal 3: Tierra

Conectores J11 y J12: Entradas disponibles en la compuerta 74HC08 (U8C)

Conector J11: Terminal 1: Entrada a U8C
Terminal 2: Vcc (+5V)
Terminal 3: Tierra

Conector J12: Terminal 1: Entrada a U8C
Terminal 2: Vcc (+5V)
Terminal 3: Tierra

Conector J13: Salida disponible de la compuerta 74HC08 (U8C)

Terminal 1: Salida de U8C
Terminal 2: Tierra

Conector J14: Entrada del microcontrolador (U1) por la terminal T1 (Reloj 1), que es una señal requerida por el microcontrolador para conectar un tacómetro a una frecuencia máxima de 250 KHz, con un ciclo de trabajo en la señal de entrada superior a los 20 μs a 5 Vpp.

Terminal 1: Señal de entrada a T1
Terminal 2: Vcc (+5V)
Terminal 3: Tierra

Conector J15: Entrada de U1 por la terminal T0 (Reloj 0), la cual es una entrada disponible para conectar un contador, pero sin uso por el momento).

Terminal 1: Señal de entrada a T0
Terminal 2: Vcc (+5V)
Terminal 3: Tierra

Conector J16: Entrada de U1 por la terminal INT0 (Interrupción externa 0), la cual se utiliza para los paros de emergencia (señal TTL de más alta prioridad).

Terminal 1: Señal de entrada a INT0
Terminal 2: Vcc (+5V)
Terminal 3: Tierra

Conector J17: Polarización de la Tarjeta Principal

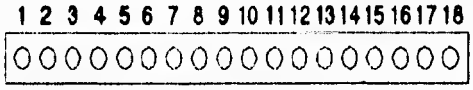
Terminal 1: Polarización con la Batería de +12V
Terminal 2: Tierra

Conector J8: Señales utilizadas para la comunicación con la PC.

Terminal 1: Tx
Terminal 2: Rx
Terminal 3: Tierra

Las señales de salida a los motores de pasos se obtienen de los conectores M1-M6 situados en la Etapa de Potencia 1 y 2. (Para ubicación consultar el Apéndice D y E respectivamente).

M1: Motor 1 (Muñeca 1)
M2: Motor 2 (Muñeca 2)
M3: Motor 3 (Base)
M4: Motor 4 (Hombro)
M5: Motor 5 (Tenaza)
M6: Motor 6 (Codo)



Terminales 1 y 2: Tierra
Terminales 3, 4, 9 y 14: Polarización (+12 V)

Terminal 5: A0
Terminal 6: A1
Terminal 7: A2
Terminal 8: A3
— M1

Terminal 10: C4
Terminal 11: C5
Terminal 12: C6
Terminal 13: C7
— M6

Terminal 15: A4
Terminal 16: A5
Terminal 17: A6
Terminal 18: A7
— M2

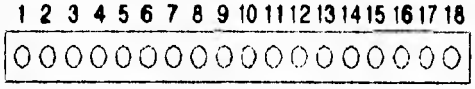
Terminales 1 y 2: Tierra
Terminales 3, 4, 9 y 14: Polarización (+12 V)

Terminal 5: B4
Terminal 6: B5
Terminal 7: B6
Terminal 8: B7
— M4

Terminal 10: C0
Terminal 11: C1
Terminal 12: C2
Terminal 13: C3
— M5

Terminal 15: B0
Terminal 16: B1
Terminal 17: B2
Terminal 18: B3
— M3

Con respecto a la Etapa de Potencia 3 (Apéndice F), presenta salidas para dos motores de pasos y dos motores de DC, como se muestra a continuación:



Terminales 1 y 2: Tierra
Terminales 3 y 4: Polarización (+6 Volts)
Terminales 9 y 14: Polarización (+12 Volts)
Terminales 5 y 6: Conexión para el Motor 1 de DC
Terminales 7 y 8: Conexión para el Motor 2 de DC

Terminal 10: C0 ┐
Terminal 11: C1 ├ M5
Terminal 12: C2 │
Terminal 13: C3 ┘

Terminal 15: B0 ┐
Terminal 16: B1 ├ M3
Terminal 17: B2 │
Terminal 18: B3 ┘

Una vez conectado el sistema, grabado el programa monitor en la EPROM del 8751H, se puede proceder a encender el sistema mediante el interruptor externo. (En caso de querer inicializar el sistema manualmente se debe presionar el switch S1).

Para verificar el correcto funcionamiento del sistema se deben seguir lo siguientes pasos:

1. Ejecutar el programa de control llamado APRENDER.EXE (Apéndice B). (Turbo Pascal versión 6.0). El que presentará el siguiente menú de acciones:

F1 --> GIRAR BASE (MOTOR 3)
F2 --> MOVER CODO (MOTOR 6)
F3 --> MOVER HOMBRO (MOTOR 4)
F4 --> MOVER TENAZA (MOTOR 5)
F5 --> GIRAR MANO (MOTOR 1,2)
F6 --> MOVER MANO (MOTOR 1,2)
F7 --> MOVER BASE Y CODO
F8 --> MOVER HOMBRO Y BASE
F9 --> MOVER HOMBRO Y CODO
F10 -> MOVER HOMBRO, BASE Y CODO
SHIFT-F1 REPETIR RUTINA APRENDIDA
SHIFT-F2 EJECUTAR RUTINA GRABADA
L --> LIMPIAR ARCHIVO
R --> RECIBIR DATOS DE LA SONDA
I --> REGRESAR AL PUNTO DE INICIO
ESC -> SALIR
BARRA PARO DE EMERGENCIA

Terminales 1 y 2: Tierra
Terminales 3 y 4: Polarización (+6 Volts)
Terminales 9 y 14: Polarización (+12 Volts)
Terminales 5 y 6: Conexión para el Motor 1 de DC
Terminales 7 y 8: Conexión para el Motor 2 de DC

Terminal 10: C0 ┐
Terminal 11: C1 ├ M5
Terminal 12: C2 │
Terminal 13: C3 ┘

Terminal 15: B0 ┐
Terminal 16: B1 ├ M3
Terminal 17: B2 │
Terminal 18: B3 ┘

Una vez conectado el sistema, grabado el programa monitor en la EPROM del 8751H, se puede proceder a encender el sistema mediante el interruptor externo. (En caso de querer inicializar el sistema manualmente se debe presionar el switch S1).

Para verificar el correcto funcionamiento del sistema se deben seguir lo siguientes pasos:

1. Ejecutar el programa de control llamado APRENDER.EXE (Apéndice B). (Turbo Pascal versión 6.0). El que presentará el siguiente menú de acciones:

F1 --> GIRAR BASE (MOTOR 3)
F2 --> MOVER CODO (MOTOR 6)
F3 --> MOVER HOMBRO (MOTOR 4)
F4 --> MOVER TENAZA (MOTOR 5)
F5 --> GIRAR MANO (MOTOR 1,2)
F6 --> MOVER MANO (MOTOR 1,2)
F7 --> MOVER BASE Y CODO
F8 --> MOVER HOMBRO Y BASE
F9 --> MOVER HOMBRO Y CODO
F10 -> MOVER HOMBRO, BASE Y CODO
SHIFT-F1 REPETIR RUTINA APRENDIDA
SHIFT-F2 EJECUTAR RUTINA GRABADA
L --> LIMPIAR ARCHIVO
R --> RECIBIR DATOS DE LA SONDA
I --> REGRESAR AL PUNTO DE INICIO
ESC -> SALIR
BARRA PARO DE EMERGENCIA

2. Desde el programa de control de la PC se envía la señal para mover la Base (M3) y el Hombro (M4) hacia adelante, movimiento que debe ejecutar el Brazo Mecánico tantas veces como se presione la tecla F8. En caso de no estar conectado el Brazo; las señales de salida que deberán tenerse en la etapa de potencia tanto en M3, como en M4 son: 1100B, 1001B, 0011B y 0110B. Esta secuencia se repetirá cada vez que se presione F8.

3. A continuación se presiona la tecla R, para activar el ADC. Las lecturas adquiridas serán guardadas en la EEPROM, para luego ser transmitidas a la PC y desplegadas en pantalla, de la siguiente forma:

TIEMPO [SEG]	DISTANCIA [m]	ORIENTACIÓN [m]	TEMPERATURA [°C]	HUMEDAD [%]	GAMA [Rad]	BETA [Rad]

NOTA: Si son ejecutadas las operaciones anteriores correctamente, damos por terminadas las pruebas de operación por lo que el Sistema está preparado para iniciar su operación, de otra manera existe la posibilidad de algún daño el cuál deberá corregirse para garantizar su correcto funcionamiento.

CAPITULO 6

PRUEBAS Y RESULTADOS

6. PRUEBAS Y RESULTADOS.

Manipulador:

Las pruebas realizadas con el Mini Brazo de Electrónica Veneta modelo RB-4/EV de seis grados de libertad han sido exitosas, lográndose describir trayectorias con buena repetibilidad y exactitud a diferentes velocidades, encontrándose que el intervalo de retardo entre una secuencia de pasos y otra (variación de la velocidad) está entre 118 y 168 ms; intervalo que asegura el rendimiento óptimo del manipulador. La velocidad programada en el microcontrolador 8751H es fija (140 ms), lo que no representa ninguna limitación cuando el control se realiza con la PC, debido a que se puede controlar la velocidad del manipulador tan sólo con variar el retardo que solicita el Programa de Aprendizaje. Con respecto a las rutinas disponibles grabadas en el microcontrolador, pueden ejecutar los movimientos del brazo a diferentes velocidades, claro esta que dentro del rango establecido previamente. En cuanto al control a distancia, la velocidad es constante, pero con la posibilidad de tener diferentes velocidades de operación, cantidad que esta en función del número de teclas disponibles en el Control Remoto.

Plataforma:

En cuanto al sistema de desplazamiento con los motores de D.C. alimentado con baterías recargables el control se realiza de dos formas diferentes, una para la tracción y otra para la dirección, esta diferencia se debe a que los requerimientos son diferentes para las dos formas de desplazamiento.

En la tracción se esta utilizando un control de ciclo abierto, por lo que no se tiene una señal de referencia, simplemente el motor tiene dos estados, encendido o apagado. Con este tipo de control se puede obtener el par máximo del motor, lo que redunda en la posibilidad de mover hasta unos 15 Kg.

Mientras que para la dirección se observo que era indispensable tener mucha precisión razón por la cual el control se realiza por medio del ancho del pulso (aproximadamente de 0.8 segundos), lo que implica que para poder girar la rueda se debe oprimir la tecla correspondiente hasta llegar a la posición deseada, para la tracción sólo es necesario presionar una sola vez la tecla de avance o retroceso hasta alcanzar la posición deseada, momento en el que se le indicará a la Sonda la siguiente operación a ejecutar.

Adquisición y Manejo de Datos:

Para el monitoreo, el sistema presenta un alto rendimiento en las pruebas de adquisición de datos, acción que toma alrededor de 100 microsegundos, con una capacidad de almacenamiento de 8191 datos de 8 bits; en cuanto a la transmisión de los mismos a la PC para su análisis, se efectúa a 1200 bauds/seg.

Suministro de Energía:

En cuanto al consumo de energía del sistema, la etapa digital opera con 270 mA, cada motor de pasos del brazo mecánico (seis en total) consume 0.8 Ampers polarizados con 12 V, mientras que el motor de la tracción consume 2.14 Ampers y el de dirección 1.11 Ampers polarizados con 6 volts.

El sistema cuenta con dos baterías recargables, una de 12 volts (7 Ah) con la que se polarizan la etapa digital y los motores de pasos del manipulador; con lo que a plena carga se consume un total de 5.07 Ampers, lo que implica que la batería puede suministrar energía aproximadamente por una hora 38 minutos bajo estas condiciones; pero debido a que el manipulador solo opera en la zona de muestreo, el tiempo que consume energía de la batería es mínimo. La batería de 6 volts (10 Ah) polariza el motor de tracción y dirección por lo que a plena carga se consume un total de 3.25 A, con lo que se tiene que la batería puede suministrar energía durante 3 horas.

CAPITULO 7

CONCLUSIONES

CONCLUSIONES

Al inicio de este trabajo se plantearon ciertos objetivos a alcanzar como son el diseño y construcción del control de movimiento, del sistema de adquisición de los parámetros críticos de una planta nuclear y la realización de las rutinas de muestreo por medio de un manipulador de seis grados de libertad, los cuales se lograron satisfactoriamente e incluso se superaron, puesto que el sistema además de contar con un medio de almacenamiento de las lecturas adquiridas, puede transmitirlas a una PC para su análisis con la ayuda de un Sistema de Transmisión-Recepción compatible con el estándar RS-232. Debido a las mismas exigencias requeridas para utilizar éste sistema en una zona con ambiente radioactivo (meta que se quiere alcanzar) se contemplo en el diseño final la conexión a futuro de un sistema de localización para determinar la posición exacta del vehículo, de un sistema de paros de emergencia para evitar colisiones y de un sistema de transmisión de video para conocer el área que se esta inspeccionando, pues es indispensable lograr un alto rendimiento del sistema, que de lo contrario podría incrementar los riesgos en lugar de aminorarlos si se llegara a perder el control de la Sonda Robot.

Debido a los resultados obtenidos, podemos afirmar que el prototipo realizado ofrece una buena perspectiva a futuro para la posible fabricación del mismo y desde el punto de vista de la ingeniería el costo resulta sumamente bajo (alrededor de los 900 dolares, sin incluir el brazo mecánico y los sensores) si lo comparamos con el de un Robot comercial que se cotiza en unos 10,000 dolares; mientras qué el beneficio obtenido resulta en la reducción de los costos de operación en la supervisión y mantenimiento de la planta y sobre todo en un beneficio directo al personal de protección, ya que evitará la necesidad de exponer al personal por contar con dispositivos autónomos.

El sistema podría utilizarse prácticamente en cualquier tipo de ambiente realizando modificaciones únicamente al mecanismo de desplazamiento y tomando las medidas adecuadas para el tipo de condiciones prevalecientes en el medio. Por otro lado el sistema se diseño modularmente, lo que otorga beneficios adicionales como es la posibilidad de reemplazar alguna etapa para poder dar un giro a la aplicación del sistema y realizar el control de un periférico distinto al utilizado en este prototipo.

Durante el desarrollo de este trabajo se fué creando paralelamente un sistema de aprendizaje como se mencionó en el capítulo cuatro, que otorga muchas ventajas en aplicaciones de robótica sobre todo si se considera que es necesario generar rutinas para evitar obstáculos y las dificultades que esto implica. Las ventajas que presenta son la posibilidad de crear y guardar rutinas, ejecutar rutinas a diferentes velocidades y modificarlas para mejorarlas o complementar algunas otras; acciones que se realizan de manera sencilla. Si se construye un modelo real de la zona que se va a explorar o en su defecto creando los algoritmos de evasión de obstáculos en el lugar de operación cuando sea posible el rendimiento se incrementaría notablemente.

APÉNDICE A

PROGRAMA PARA EL MICROCONTROLADOR 8751H

El siguiente programa permite realizar una prueba para determinar si todos los periféricos del sistema trabajan adecuadamente, así como permitirle al operador controlar la Sonda.

Apéndice A. Programa Monitor.

```
CSEG  ;INICIO DE PROGRAMA
ORG 00H
LJMP INICIO
ORG 13H
LJMP EXT1
ORG 23H
LJMP SERIE

ORG 32H
INICIO: MOV SP,#20H
        mov psw,#00h
        MOV SCON,#50H ;INICIALIZA PUERTO SERIE PARA
                      8-BIT UART
                ;SCON=01010000B
        MOV TMOD,#25H  ;TIMER 0 EN MODO 2 y
                ;TIMER 1 EN MODO 1 TMOD=0010
                      0101B
        MOV TL0,#0F0H
        mov TH0,#00h
        MOV TH1,#0F3H  ;BAUDAJE DE 1.2k EN MODO 3
                      (REGISTRO)
        MOV TCON,#55H  ;TIMER 1 y 0 ENCENDIDOS
                ;INTERRUPCION 0 TRABAJANDO CON
                      FLANCOS
        MOV IE,#10010110B ;HABILITACION DE
                      INTERRUPCIONES
        MOV IP,#00H      ;HABILITACION PARA
                      SELECCIONAR LA
                ;PRIORIDAD DE CADA INTERRUPCION
        MOV R5,#00H
        mov r4,#80h
        mov r3,#00h
        MOV R2,#00H
        mov p2,#0A0h
        MOV A,#80H
        MOV DPTR,#2003H    ;PROGRAMANDO 82C55
        MOVX @DPTR,A
        MOV A,#00H
        mov dptr,#0a000h   ;DESACTIVAR PERIFERICOS
        movx @dptr,a

ESPERA: MOV DPTR,#2000H   ;INICIO DE RUTINA DE
                      ESPERA
        MOVX @DPTR,A
        MOV DPTR,#2001H
        MOVX @DPTR,A
        mov dptr,#2002h
        movx @dptr,a
        jb f0,espera
FALTAN: JB RB8,SIGUE
        SJMP ESPERA
SIGUE: CLR RB8
        SJMP FALTAN   ;FIN DE RUTINA DE ESPERA


;SUBRUTINAS

  EXT1: PUSH ACC  ;INICIO DE RUTINA DE SERVICIO
        push dph  ;PARA INTERRUPCION EXTERNA 1
        push dpl
        mov dptr,#0E00Fh   ;leyendo ADC canal 16
        movx a,@dptr
        mov b,a
        mov dph,r4        ;ACTIVANDO EEPROM
        mov dpl,r3
GUARDA: movX @DPTR,A
        inc r3
        cjne r3,#00H,libre
        inc r4
libre:  CJNE R4,#9FH,LIBRE1
        CJNE R3,#0FFH,LIBRE1
        MOV R6,R4
        MOV R7,R3
        MOV R3,#00H
        MOV R4,#80H
        SETB TI
LIBRE1: pop dpl
        pop dph
        POP ACC
        RETI    ;FIN DE RUTINA DE SERVICIO EXT1

 SERIE: PUSH PSW    ;INICIO DE RUTINA DE SERVICIO
        PUSH ACC    ;PARA INTERRUPCION SERIE
        PUSH DPL
        PUSH DPH
        MOV A,SCON
        ANL A,#03H
        DEC A
        JNZ PS_OUT
 PS_IN: CLR RI
        MOV A,SBUF      ;RUTINA PARA RECEPCION DE
                      DATOS
        jb f0,fips
        lcall analizo
        MOV A,#00H
        MOV DPTR,#2000H
        MOVX @DPTR,A
        mov dptr,#2002h
        movx @dptr,a
        SJMP FIPS     ;FIN DE RUTINA PARA RECIBIR
                      DATOS

 PS_OUT: CLR TI   ;RUTINA PARA TRANSMITIR DATOS
 EPROM: mov sbuf,r6
        jnb ti,$
        clr ti
        mov sbuf,r7
        jnb ti,$
        clr ti
        cjne r6,#80h,continua
        cjne r7,#00,continua
        ljmp fips
continua:mov a,#00h
        mov dptr,#8000h
     T: push acc
        movx a,@dptr
        mov sbuf,a
        jnb ti,$
```

```
      clr ti
      inc dptr
      pop acc
      inc acc
      cjne a,r7,T
      PUSH ACC
      MOV A,R6
      CJNE A,DPH,AUMENTA
      POP ACC
      SJMP FIPS
AUMENTA: POP ACC
      SJMP T  ;FIN DE RUTINA PARA TRANSMITIR
                  OATOS
 FIPS:  POP DPH
      POP DPL
      POP ACC
      POP PSW
      RETI              ;FIN DE INTERRUPCION SERIE

analiza: CJNE a,#31H,PASO2   ;INICIO DE RUTINA PARA
                             SELECCIONAR ACCION A
                             EJECUTAR
      MOV DPTR,#2002H
      MOV A,#0c0h
      MOVX @DPTR,A       ;Base y Codo hacia Adelante
      MOV DPTR,#2001H
      MOV A,#0cH
      MOVX @DPTR,A
      lcall retardos
      MDV DPTR,#2002H
      mov A,#90h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#09H
      MOVX @DPTR,A
      lcall retardos
      MOV DPTR,#2002H
      mov A,#30h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#03H
      MOVX @DPTR,A
      lcall retardos
      MOV DPTR,#2002H
      mov A,#60h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#06H
      MOVX @DPTR,A
      lcall RETARDOS
      MOV A,#00H
      MOVX @DPTR,A
      ljmp fin
 paso2: cjne a,#32h,paso3   ;BASE Y CODO HACIA
                              ATRAS
      MOV DPTR,#2002H
      MOV A,#60h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#06H
      MOVX @DPTR,A
      lcall retardos
      MOV DPTR,#2002H
      mov A,#30h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#03H
      MOVX @DPTR,A
      lcall retardos
      MOV DPTR,#2002H
      mov A,#90h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#09H
      MOVX @DPTR,A
      lcall retardos
      MOV DPTR,#2002H
      mov A,#0c0h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#0cH
      MOVX @DPTR,A
      lcall RETARDOS
      MOV A,#00H
      MOVX @DPTR,A
      ljmp fin
PASO3: CJNE A,#33H,PASO4      ;CODO Y HOMBRO
                               HACIA ADELANTE
      MOV DPTR,#2002H
      MOV A,#0c0h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#60H
      MOVX @DPTR,A
      lcall retardos
      MOV DPTR,#2002H
      mov A,#90h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#30H
      MOVX @DPTR,A
      lcall retardos
      MOV DPTR,#2002H
      mov A,#30h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#90H
      MOVX @DPTR,A
      lcall retardos
      MOV DPTR,#2002H
      mov A,#60h
      MOVX @DPTR,A
      MOV DPTR,#2001H
      MOV A,#0c0H
      MOVX @DPTR,A
      lcall RETARDOS
      MOV A,#00H
      MOVX @DPTR,A
      ljmp fin
paso4: cjne a,#34h,paso5     ;BASE EN REVERSA
      MOV DPTR,#2001H
      mov A,#06h
      MOVX @DPTR,A
      lcall retardos
```

```
    mov A,#03h
    MOVX @DPTR,A
    lcall retardos
    mov A,#09h
    MOVX @DPTR,A
    lcall retardos
    mov A,#0ch
    MOVX @DPTR,A
    lcall retardos
    MOV A,#00H
    MOVX @DPTR,A
    ljmp fin
PASO5: CJNE A,#35H,PASO6      ;CODO EN REVERSA
    MOV DPTR,#2002H
    mov A,#60h
    MOVX @DPTR,A
    lcall retardos
    mov A,#30h
    MOVX @DPTR,A
    lcall retardos
    mov A,#90h
    MOVX @DPTR,A
    lcall retardos
    mov A,#0c0h
    MOVX @DPTR,A
    lcall retardos
    ljmp fin
paso6: cjne a,#36h,paso7      ;HOMBRO hacia adelante
    MOV DPTR,#2001H
    mov A,#60h
    MOVX @DPTR,A
    lcall retardos
    mov A,#30h
    MOVX @DPTR,A
    lcall retardos
    mov A,#90h
    MOVX @DPTR,A
    lcall retardos
    mov A,#0c0h
    MOVX @DPTR,A
    lcall retardos
    MOV A,#00H
    MOVX @DPTR,A
    ljmp fin
paso7: cjne a,#37h,paso8      ;BASE HACIA ADELANTE
    MOV DPTR,#2001H
    mov A,#0ch
    MOVX @DPTR,A
    lcall retardos
    mov A,#09h
    MOVX @DPTR,A
    lcall retardos
    mov A,#03h
    MOVX @DPTR,A
    lcall retardos
    mov A,#06h
    MOVX @DPTR,A
    lcall retardos
    MOV A,#00H
    MOVX @DPTR,A
    ljmp fin
paso8: cjne a,#38h,paso9      ;CODO HACIA ADELANTE
    MOV DPTR,#2002H
    mov A,#0c0h
    MOVX @DPTR,A
    lcall retardos
    mov A,#90h
    MOVX @DPTR,A
    lcall retardos
    mov A,#30h
    MOVX @DPTR,A
    lcall retardos
    mov A,#60h
    MOVX @DPTR,A
    lcall retardos
    ljmp fin
paso9: cjne a,#39h,paso10     ;HOMBRO en reversa
    MOV DPTR,#2001H
    mov A,#0c0h
    MOVX @DPTR,A
    lcall retardos
    mov A,#90h
    MOVX @DPTR,A
    lcall retardos
    mov A,#30h
    MOVX @DPTR,A
    lcall retardos
    mov A,#60h
    MOVX @DPTR,A
    lcall retardos
    MOV A,#00H
    MOVX @DPTR,A
    ljmp fin
PASO10: CJNE A,#40H,paso11    ;BASE, CODO Y
                                HOMBRO ADELANTE
    MOV DPTR,#2002H
    mov A,#0c0h
    MOVX @DPTR,A
    MOV DPTR,#2001H
    mov A,#6ch
    MOVX @DPTR,A
    lcall retardos
    MOV DPTR,#2002H
    mov A,#90h
    MOVX @DPTR,A
    MOV DPTR,#2001H
    mov A,#39h
    MOVX @DPTR,A
    lcall retardos
    MOV DPTR,#2002H
    mov A,#30h
    MOVX @DPTR,A
    MOV DPTR,#2001H
    mov A,#93h
    MOVX @DPTR,A
    lcall retardos
    MOV DPTR,#2002H
    mov A,#60h
    MOVX @DPTR,A
    MOV DPTR,#2001H
    mov A,#0C6h
    MOVX @DPTR,A
    lcall retardos
    MOV A,#00H
```

```
        MOVX @DPTR,A
        ljmp fin
PASO11: CJNE A,#41H,paso12    ;BASE, CODO Y
                                HOMBRO  EN
REVERSA
        MOV DPTR,#2002H
        mov A,#60h
        MOVX @DPTR,A
        MOV DPTR,#2001H
        mov A,#0c6h
        MOVX @DPTR,A
        lcall retardos
        MOV DPTR,#2002H
        mov A,#30h
        MOVX @DPTR,A
        MOV DPTR,#2001H
        mov A,#93h
        MOVX @DPTR,A
        lcall retardos
        MOV DPTR,#2002H
        mov A,#90h
        MOVX @DPTR,A
        MOV DPTR,#2001H
        mov A,#39h
        MOVX @DPTR,A
        lcall retardos
        MOV DPTR,#2002H
        mov A,#0c0h
        MOVX @DPTR,A
        MOV DPTR,#2001H
        mov A,#6ch
        MOVX @DPTR,A
        lcall retardos
        MOV A,#00H
        MOVX @DPTR,A
        ljmp fin
PASO12: CJNE A,#42H,paso13    ;MANO EN REVERSA
        MOV DPTR,#2002H
        mov A,#06h
        MOVX @DPTR,A
        lcall retardos
        mov A,#03h
        MOVX @DPTR,A
        lcall retardos
        mov A,#09h
        MOVX @DPTR,A
        lcall retardos
        mov A,#0ch
        MOVX @DPTR,A
        lcall retardos
        ljmp fin
PASO13: CJNE A,#43H,paso14    ;MANO HACIA
                                ADELANTE
        MOV DPTR,#2002H
        mov A,#0ch
        MOVX @DPTR,A
        lcall retardos
        mov A,#09h
        MOVX @DPTR,A
        lcall retardos
        mov A,#03h
        MOVX @DPTR,A
        lcall retardos
        mov A,#06h
        MOVX @DPTR,A
        lcall retardos
        ljmp fin
PASO14: CJNE A,#46H,paso15    ;codo y hombro en
                                reversa
        MOV DPTR,#2002H
        mov A,#60h
        MOVX @DPTR,A
        MOV DPTR,#2001H
        mov A,#0c0h
        MOVX @DPTR,A
        lcall retardos
        MOV DPTR,#2002H
        mov A,#30h
        MOVX @DPTR,A
        MOV DPTR,#2001H
        mov A,#90h
        MOVX @DPTR,A
        lcall retardos
        MOV DPTR,#2002H
        mov A,#90h
        MOVX @DPTR,A
        MOV DPTR,#2001H
        mov A,#30h
        MOVX @DPTR,A
        lcall retardos
        MOV DPTR,#2002H
        mov A,#0c0h
        MOVX @DPTR,A
        MOV DPTR,#2001H
        mov A,#60h
        MOVX @DPTR,A
        lcall retardos
        MOV A,#00H
        MOVX @DPTR,A
        ljmp fin
paso15: cjne a,#47h,paso16    ; Base y hombro HACIA
                                ADELANTE
        MOV DPTR,#2001H
        mov A,#6ch
        MOVX @DPTR,A
        lcall retardos
        mov A,#39h
        MOVX @DPTR,A
        lcall retardos
        mov A,#93h
        MOVX @DPTR,A
        lcall retardos
        mov A,#0c6h
        MOVX @DPTR,A
        lcall retardos
        MOV A,#00H
        MOVX @DPTR,A
        ljmp fin
paso16: CJNE A,#49H,paso17    ;Base y Hombro HACIA
                                Atras
        MOV DPTR,#2001H
        mov A,#0c6h
        MOVX @DPTR,A
        lcall retardos
```

```
      mov A,#93h
      MOVX @DPTR,A
      lcall retardos
      mov A,#39h
      MOVX @DPTR,A
      lcall retardos
      mov A,#6ch
      MOVX @DPTR,A
      lcall retardos
      MOV A,#00H
      MOVX @DPTR,A
      ljmp fin
PASO17: CJNE A,#50H,paso18    ;tracción adelante
                                (CONTINUO)
      mov OPTR,#2001H
      mov a,#00h
      movx @dptr,a
      MOV A,#20h
      MOVX @DPTR,A
      ljmp fin
PASO18: CJNE A,#51H,paso19    ;tracción atras
                                (continuo)
      mov DPTR,#2001H
      mov a,#00h
      movx @dptr,a
      MOV A,#10h
      MOVX @DPTR,A
      ljmp fin
paso19: cjne a,#52h,paso20    ;traccion adelante
                                (continuo)
      mov DPTR,#2001H      ;DIRECCION NORESTE
      MOV A,#0A0h
      MOVX @DPTR,A
      lcall tiempo
      mov a,#10h
      MOVX @DPTR,A
      ljmp fin
paso20: CJNE A,#53H,paso21    ;TRACCION ADELANTE
                                (continuo)
      mov OPTR,#2001H      ;OIRECCION NOROESTE
      MOV A,#60h
      MOVX @DPTR,A
      lcall tiempo
      MOV A,#20H
      MOVX @DPTR,A
      ljmp fin
PASO21: CJNE A,#54H,paso22    ;TRACCION ATRAS
                                (continuo)
      mov OPTR,#2001H      ;DIRECCION NORESTE
      MOV A,#90h
      MOVX @DPTR,A
      lcall tiempo
      MOV A,#10H
      MOVX @DPTR,A
      ljmp fin
PASO22: CJNE A,#55H,paso23    ;TRACCION ATRAS
                                (continuo)
      mov DPTR,#2001H      ;OIRECCION NOROESTE
      MOV A,#50h
      MOVX @DPTR,A
      lcall tiempo
      MOV A,#10H
      MOVX @DPTR,A
      ljmp fin
PASO23: CJNE A,#56h,PASO24    ;DIRECCION
    ANTIHORARIO
      MOV A,#40h
      MOVX @DPTR,A
      lcall tiempo
      MOV A,#00H
      MOVX @OPTR,A
      ljmp fin
paso24: CJNE A,#57H,paso25    ;direccion horario
      mov DPTR,#2001H
      MOV A,#80h
      MOVX @DPTR,A
      lcall tiempo
      MOV A,#00H
      MOVX @OPTR,A
      ljmp fin
PASO25: CJNE A,#58H,PASO26    ;PARO TOTAL
      MOV OPTR,#2001H
      MOV A,#00H
      MOVX @OPTR,A
      LJMP FIN
paso26: cjne a,#59h,paso27
      MOV dptr,#0E00Fh      ;activando ADC Canal 16
      movx @dptr,a
      sjmp fin
paso27: cjne a,#60h,FIN     ;INICIAR TRANSMISION
      setb ti
  fin: ret

;SUBRUTINAS
  tiempo: acall retardos
        acall retardos
        acall retardos
        acall retardos
        acall retardos
        ret

retardos: acall retardo
        acall retardo
        acall retardo
        ret

RETARDO: MOV A,#37H     ;RUTINA OE RETARDO FIJO
SALTO:   NOP
      DEC A
      PUSH ACC
      ACALL RETARDO1
      POP ACC
      JNZ SALTO
      RET

RETARDO1: MOV A,#0FH
SALTO1:   NOP
      NOP
      DEC A
      JNZ SALTO1
      RET          ;FIN DE RUTINA OE RETARDO

      ENO   ;FIN DE PROGRAMA
```

APÉNDICE B

PROGRAMA PARA UNA PC EN PASCAL

El siguiente programa permite al operador controlar la Sonda desde una PC, y principalmente realizar rutinas de muestreo con el Brazo Mecánico.

Apéndice B. Programa Aprender.

```
{PROGRAMA PARA UNA COMPUTADORA PERSONAL
REALIZADO EN PASCAL, POR MEDIO
 DEL CUAL NOS COMUNICAMOS CON UN
MICROPROCESADOR, A TRAVES DEL PUERTO
 SERIE}

{$M 50000, 50000, 310000}
PROGRAM ESCRIBE_PUERTO_SERIE;
USES DOS,CRT;

VAR
 SSR:SEARCHREC;
 TAM:LONGINT;
 codigo,color,ms,T:word;
 ARCH,RUTINA1,RUTINA2:FILE OF CHAR;
 F:FILE OF INTEGER;
 rec:real;
 MAR:FILE OF CHAR;
 D,PE,c,AUX,temp:char;

RENGLON,NUM,R,I,AA,DR,DA,BR,BA,CR,CA,MR,MA,HR,
HA,TR,TA,RR:integer;

cont,error,e,B,m1,m2,h1,h2,sa,sr,NAME,PSER,sal,O,x,s,n
,W,y:INTEGER;
 REGS : REGISTERS;
 total,retardo,opcion,ST,NOMBRE,NOMBREC:STRING[12];

 PROCEDURE INICIALIZA_BAUDAJE(PSER:INTEGER);

 {1200:  BAUD_RATE:=$83;  {*10000011b   =1200
BAUDS, SIN PARIDAD,
  1 BIT stop, 8 BITS CARACTER *)}

  BEGIN
  WITH REGS DO
    BEGIN
     DX:=PSER;    {COM1}
     AH:=0;    {SERVICIO 0 INICIALIZAR PARAMETROS
DEL PUERTO SERIE}
                  A  L  :  =  $  8  3  ;
{BAUDAJE,PARIDAD,STOP_BITS,FORMATO PALABRA}
     INTR(20,REGS);   {SERVICIO DE COMUNICACION
SERIAL ROM-BIOS}
    END;
 END;

 PROCEDURE RESETEA_PTO_SERIAL(PSER:INTEGER);
  BEGIN
    WITH REGS DO
      BEGIN
        AH:=2;
        DX:=PSER;       {COM1}
        INTR(20,REGS);   {RECIBIR UN CARACTER POR
PUERTO SERIE}
      END;
  END;

 PROCEDURE ENVIA_DATO(PSER:INTEGER;DATO:BYTE);
  BEGIN
    WITH REGS DO
      BEGIN
        AH:=1;       {ENVIAR CARACTER}
        AL:=DATO;   {DATO A TRANSMITIR}
        DX:=PSER;       {COM1}
        INTR(20,REGS);   {ENVIAR AL PUERTO SERIE}
    END;
  END;


function digito(N:Byte):char;
begin
  if n<10 then
    digito:=chr(ord('0')+n)
  else
    digito:=chr(ord('A')+(n-10))
end;

procedure ConHex(num:byte);
var x,y:char;
begin
  x:=digito(Lo(num) div 16);
  y:=digito(Lo(num)mod 16);
  write(RUTINA2,x);
  write(RUTINA2,y);
end;

procedure declmal(c:char);
  begin
    if c='A' then TOTAL:=CONCAT(TOTAL,'10')
    else if c='B' then TOTAL:=CONCAT(TOTAL,'11')
    else if c='C' then TOTAL:=CONCAT(TOTAL,'12')
    else if c='D' then TOTAL:=CONCAT(TOTAL,'13')
    else if c='E' then TOTAL:=CONCAT(TOTAL,'14')
    else if c='F' then TOTAL:=CONCAT(TOTAL,'15')
    ELSE TOTAL:=CONCAT(TOTAL,C);
  END;


 PROCEDURE RECIBE_DATO(PSER:INTEGER);
  begin
    with regs do
      begin
        ah:=2;
        dx:=pser;
        intr(20,regs);
        num:=AL;
      end;
  end;

 PROCEDURE OCHO;
  begin
    READ(RUTINA2,C);
    WRITE(C);
    READ(RUTINA2,C);
    WRITE(C);
```

Apéndice B. Programa Aprender.

```
{PROGRAMA PARA UNA COMPUTADORA PERSONAL
REALIZADO EN PASCAL, POR MEDIO
 DEL CUAL NOS COMUNICAMOS CON UN
MICROPROCESADOR, A TRAVES DEL PUERTO
 SERIE}

{$M 50000, 50000, 310000}
PROGRAM ESCRIBE_PUERTO_SERIE;
USES DOS,CRT;

VAR
 SSR:SEARCHREC;
 TAM:LONGINT;
 codigo,color,ms,T:word;
 ARCH,RUTINA1,RUTINA2:FILE OF CHAR;
 F:FILE OF INTEGER;
 rec:real;
 MAR:FILE OF CHAR;
 D,PE,c,AUX,temp:char;

RENGLON,NUM,R,I,AA,DR,DA,BR,BA,CR,CA,MR,MA,HR,
HA,TR,TA,RR:integer;

cont,error,e,B,m1,m2,h1,h2,sa,sr,NAME,PSER,sal,Q,x,s,n
,W,y:INTEGER;
 REGS : REGISTERS;
 total,retardo,opcion,ST,NOMBRE,NOMBREC:STRING[12];

 PROCEDURE INICIALIZA_BAUDAJE(PSER:INTEGER);

 {1200:   BAUD_RATE:=$83;   {*10000011b   =1200
BAUDS, SIN PARIDAD,
  1 BIT stop, 8 BITS CARACTER *)}

 BEGIN
 WITH REGS DO
   BEGIN
    DX:=PSER;    {COM1}
    AH:=0;     {SERVICIO 0 INICIALIZAR PARAMETROS
DEL PUERTO SERIE}
                    A  L  :  =  $  8  3  ;
{BAUDAJE,PARIDAD,STOP_BITS,FORMATO PALABRA}
    INTR(20,REGS);    {SERVICIO DE COMUNICACION
SERIAL ROM-BIOS}
   END;
 END;

 PROCEDURE RESETEA_PTO_SERIAL(PSER:INTEGER);
  BEGIN
    WITH REGS DO
      BEGIN
        AH:=2;
        DX:=PSER;       {COM1}
        INTR(20,REGS);   {RECIBIR UN CARACTER POR
PUERTO SERIE}
      END;
  END;

 PROCEDURE ENVIA_DATO(PSER:INTEGER;DATO:BYTE);
  BEGIN
    WITH REGS DO
      BEGIN
        AH:=1;       {ENVIAR CARACTER}
        AL:=DATO;    {DATO A TRANSMITIR}
        DX:=PSER;       {COM1}
        INTR(20,REGS);   {ENVIAR AL PUERTO SERIE}
    END;
  END;


function digito(N:Byte):char;
begin
  if n<10 then
    digito:=chr(ord('0')+n)
  else
    digito:=chr(ord('A')+(n-10))
end;

procedure ConHex(num:byte);
var x,y:char;
begin
  x:=digito(Lo(num) div 16);
  y:=digito(Lo(num)mod 16);
  write(RUTINA2,x);
  write(RUTINA2,y);
end;

procedure decimal(c:char);
  begin
    if c='A' then TOTAL:=CONCAT(TOTAL,'10')
    else if c='B' then TOTAL:=CONCAT(TOTAL,'11')
    else if c='C' then TOTAL:=CONCAT(TOTAL,'12')
    else if c='D' then TOTAL:=CONCAT(TOTAL,'13')
    else if c='E' then TOTAL:=CONCAT(TOTAL,'14')
    else if c='F' then TOTAL:=CONCAT(TOTAL,'15')
    ELSE TOTAL:=CONCAT(TOTAL,C);
  END;


PROCEDURE RECIBE_DATO(PSER:INTEGER);
  begin
    with regs do
      begin
        ah:=2;
        dx:=pser;
        intr(20,regs);
        num:=AL;
      end;
  end;

PROCEDURE DCHD;
  begin
    READ(RUTINA2,C);
    WRITE(C);
    READ(RUTINA2,C);
    WRITE(C);
```

```
  END;

  procedure blanco;
    var b:integer;
    begin
      for b:=1 to 30 do
      write(' ');
    end;

  procedure sign;
    var Z:integer;
    begin
      TEXTBACKGROUND(0);
      W:=wherex;
      y:=wherey;
      window(4,20,36,25);
      highvideo;
      TEXTBACKGRDUND(1);
      write(#201,#205,#205,#205,#205,#205,#205,#205);
                 w r i t e l n ( ' E s t a d o   d e l
Brazo',#205,#205,#205,#205,#205,#205,#187);
      for Z:=1 to 4 do
        begin
          write(#186);
          blanco;
          writeln(#186);
        end;

write(#200,#205,#205,#205,#205,#205,#205,#205,#2
05,#205,#205,#205,#205,#205,#205,#205);

write(#205,#205,#205,#205,#205,#205,#205,#205,#2
05,#205,#205,#205,#205,#205,#205,#188);
      window(8,22,36,25);
      textbackground(11);
      highvideo;
    end;
  procedure rest;
    begin
      delay(ms);
      window(38,1,80,24);
      gotoxy(W,y);
      textbackground(10);
    end;
  procedure rest1;
    begin
      window(1,1,36,17);
      lowvideo;
      gotoxy(W,y);
      textbackground(0);
    end;

  PROCEDURE REPETIR(C:CHAR);
    BEGIN
      if c=#49 then
        begin
          envia_dato(PSER,$32);
          write(' ','DA');
          sign;
          writeln(' Base Sentido Horario ');
          write('  Codo hacia Adelante  ');
          rest;
          DR:=DR+1;
        end
      else if c=#50 then
        begin
          envia_dato(PSER,$31);
          write(' ','DR');
          sign;
          writeln(' Base Sentido Antihorario ');
          write('      Codo en Reversa      ');
          rest;
          DA:=DA+1;
        end
      else if c=#51 then
        begin
          envia_dato(PSER,$33);
          write(' ','RR');
          sign;
          writeln('  Codo en Reversa  ');
          write(' Hombro en Reversa ');
          rest;
          AA:=AA+1;
        end
      else if c=#52 then
        begin
          envia_dato(PSER,$37);
          write(' ','BR');
          sign;
          write(' Base Sentido Antihorario ');
          rest;
          BA:=BA+1;
        end
      else if c=#53 then
        begin
          envia_dato(PSER,$38);
          write(' ','CR');
          sign;
          write(' Codo en Reversa ');
          rest;
          CA:=CA+1;
        end
      else if c=#54 then
        begin
          envia_dato(PSER,$36);
          write(' ','HR');
          sign;
          write(' Hombro en Reversa ');
          rest;
          HA:=HA+1;
        end
      else if c='7' then
        begin
          envia_dato(PSER,$34);
          write(' ','BA');
          sign;
          write(' Base Sentido Horario ');
          rest;
          BR:=BR+1;
        end
      else if c='8' then
        begin
          envia_dato(PSER,$35);
          write(' ','CA');
```

```
    sign;
    write(' Codo hacia Adelante ');
    rest;
    CR:=CR+1;
   end
  else if c='9' then
   begin
    envia_dato(PSER,$39);
    write(' ','HA');
    sign;
    write(' Hombro hacia Adelante ');
    rest;
    HR:=HR+1;
   end
  else if c='0' then
   begin
    envia_dato(PSER,$41);
    write(' ','TA');
    sign;
    writeln(' Base Sentido Horario ');
    writeln(' Codo hacia Adelante ');
    write(' Hombro hacia Adelante ');
    rest;
    TR:=TR+1;
   end
  else if c=#46 then
   begin
    envia_dato(PSER,$40);
    write(' ','TR');
    sign;
    writeln(' Base Sentido Antihorario ');
    writeln('   Codo en Reversa      ');
    write('   Hombro en Reversa    ');
    rest;
    TA:=TA+1;
   end
  else if c=#42 then
   begin
    envia_dato(PSER,$42);
    write(' ','MR');
    sign;
    write(' Mano Giro - ');
    rest;
    MA:=MA+1;
   end
  else if c=#47 then
   begin
    envia_dato(PSER,$43);
    write(' ','MA');
    sign;
    write(' Mano Giro + ');
    rest;
    MR:=MR+1;
   end
  else if c='+' then
   begin
    envia_dato(PSER,$46);
    write(' ','AA');
    sign;
    writeln(' Codo hacia Adelante ');
    write(' Hombro hacia Adelante ');
    rest;
    RR:=RR+1;
   end
  else if c=#080 then
   begin
    envia_dato(PSER,$47);
    write(' ','1-');
    sign;
    writeln(' Base Sentido Antihorario ');
    write('   Hombro en Reversa   ');
    rest;
    m1:=m1+1;
   end
  else if c=#072 then
   begin
    envia_dato(PSER,$49);
    write(' ','1+');
    sign;
    writeln(' Base Sentido Horario ');
    write(' Hombro hacia Adelante ');
    rest;
    h1:=h1+1;
   end
  else if c=#077 then
   begin
    envia_dato(PSER,$50);
    write(' ','G+');
    sign;
    write(' Giro de Mano + ');
    rest;
    m2:=m2+1;
   end
  else if c=#075 then
   begin
    envia_dato(PSER,$51);
    write(' ','2+');
    sign;
    write(' Giro de Mano - ');
    rest;
    h2:=h2+1;
   end
  else if c=#45 then
   begin
    envia_dato(PSER,$52);
    write(' ','SA');
    sign;
    write(' Todos hacia Adelante ');
    rest;
    sr:=sr+1;
   end
  else if c=#09 then
   begin
    envia_dato(PSER,$53);
    write(' ','SR');
    sign;
    write(' Todos en Reversa ');
    rest;
    sa:=sa+1;
   end
END;

PROCEDURE EJECUTA;
  BEGIN
```

```
        if PE = #32 then
        ELSE
          BEGIN
            IF KEYPRESSED THEN
              BEGIN
                PE:=READKEY;
                IF PE <> #32 THEN
                  BEGIN
                    Q:=Q+1;
                    REPETIR(C)
                  END;
              END
            ELSE
              BEGIN
                Q:=Q+1;
                REPETIR(C);
              END
          END
    END;


PROCEDURE INICIO(C:CHAR);
  BEGIN
    if c=#49 then
      begin
        envia_dato(PSER,$31);
        sign;
        writeln(' Base Sentido Antihorario ');
        write('     Codo en Reversa     ');
        rest;
      end
    else if c=#50 then
      begin
        envia_dato(PSER,$32);
        sign;
        writeln(' Base Sentido Horario ');
        write('  Codo hacia Adelante  ');
        rest;
      end
    else if c=#51 then
      begin
        envia_dato(PSER,$46);
        sign;
        writeln('  Hombro hacia Adelante  ');
        write('  Codo hacia Adelante  ');
        rest;
      end
    else if c=#52 then
      begin
        envia_dato(PSER,$34);
        sign;
        write(' Base Sentido Horario ');
        rest;
      end
    else if c=#53 then
      begin
        envia_dato(PSER,$35);
        sign;
        write(' Codo hacia Adelante ');
        rest;
      end
    else if c=#54 then
      begin
        envia_dato(PSER,$39);
        sign;
        write(' Hombro hacia Adelante ');
        rest;
      end
    else if c='7' then
      begin
        envia_dato(PSER,$37);
        sign;
        write(' Base Sentido Antihorario ');
        rest;
      end
    else if c='8' then
      begin
        envia_dato(PSER,$38);
        sign;
        write(' Codo en Reversa ');
        rest;
      end
    else if c='9' then
      begin
        envia_dato(PSER,$36);
        sign;
        write(' Hombro en Reversa ');
        rest;
      end
    else if c='0' then
      begin
        envia_dato(PSER,$40);
        sign;
        writeln(' Base Sentido Antihorario ');
        writeln('     Codo en Reversa     ');
        write('     Hombro en Reversa     ');
        rest;
      end
    else if c=#46 then
      begin
        envia_dato(PSER,$41);
        sign;
        writeln(' Base Sentido Horario ');
        writeln(' Codo hacia Adelante  ');
        write(' Hombro hacia Adelante ');
        rest;
      end
    else if c=#42 then
      begin
        envia_dato(PSER,$43);
        sign;
        write(' Mano hacia Adelante ');
        rest;
      end
    else if c=#47 then
      begin
        envia_dato(PSER,$42);
        sign;
        write(' Mano en Reversa ');
        rest;
      end
    else if c='+' then
      begin
        envia_dato(PSER,$33);
```

```
        sign;
        writeln('  Codo en Reversa  ');
        write(' Hombro en Reversa ');
        rest;
      end
    else if c = #080 then
      begin
        envia_dato(PSER,$49);
        sign;
        writeln(' Base Sentido Horario ');
        write(' Hombro hacia Adelante ');
        rest;
      end
    else if c = #077 then
      begin
        envia_dato(PSER,$51);
        sign;
        write('  Giro de Mano -  ');
        rest;
      end
    else if c = #072 then
      begin
        envia_dato(PSER,$47);
        sign;
        writeln(' Base Sentido Antihorario ');
        write('    Hombro en Reverse     ');
        rest;
      end
    else if c = #075 then
      begin
        envia_dato(PSER,$50);
        sign;
        write('  Giro de Mano  +  ');
        rest;
      end
    else if c = #45 then
      begin
        envia_dato(PSER,$53);
        sign;
        write('  Todos en Reversa  ');
        rest;
      end
    else if c = #09 then
      begin
        envia_dato(PSER,$52);
        sign;
        write('  Todos hacia Adelante  ');
        rest;
      end
  END;

PROCEDURE INI;
  BEGIN
    I:=0;DR:=0;DA:=0;SA:=0;SR:=D;
    BR:=0;BA:=0;CR:=0;CA:=0;h2:=0;m2:=0;
    MR:=0;MA:=0;RR:=0;HR:=0;h1:=0;m1:=0;
    HA:=0;TA:=0;TR:=0;AA:=0;
  END;

PROCEDURE DRIGEN;
  BEGIN
    RESET(mar);
        q:=0;
        INI;
        WHILE EOF(MAR) DO
          BEGIN
            READ(MAR,C);
            Q:=Q+1;
          END;

        Q:=Q-1;
        reset(MAR);
        PE:=#00;
        WHILE Q <> -1 DO
          BEGIN
            SEEK(MAR,Q);
            READ(MAR,C);
            IF PE= #32 THEN
              Q:=Q-1
            ELSE
              BEGIN
                IF KEYPRESSED THEN
                  BEGIN
                    PE:=READKEY;
                    IF PE <> #32 THEN
                      BEGIN
                        INICIO(C);
                        Q:=Q-1;
                      END
                    ELSE IF PE= #32 THEN
                      Q:=q-1;
                  END
                ELSE
                  BEGIN
                    INICIO(C);
                    Q:=Q-1;
                  END
              END;
          END;
  END;

PROCEDURE CERO(I:INTEGER);
  BEGIN
    PE:=#00;
    IF DA > = DR THEN
      BEGIN
        DA:=DA-DR;
        DR:=0;
      END
    ELSE
      BEGIN
        DR:=DR-DA;
        DA:=0;
      END;
    IF AA > = RR THEN
      BEGIN
        AA:=AA-RR;
        RR:=0;
      END
    ELSE
      BEGIN
        RR:=RR-AA;
        AA:=0;
      END;
```

```
      sign;
      writeln(' Codo en Reversa ');
      write(' Hombro en Reversa ');
      rest;
     end
   else if c = #080 then
    begin
     envia_dato(PSER,$49);
     sign;
     writeln(' Base Sentido Horario ');
     write(' Hombro hacia Adelante ');
     rest;
    end
   else if c = #077 then
    begin
     envia_dato(PSER,$51);
     sign;
     write(' Giro de Mano - ');
     rest;
    end
   else if c = #072 then
    begin
     envia_dato(PSER,$47);
     sign;
     writeln(' Base Sentido Antihorario ');
     write('  Hombro en Reversa  ');
     rest;
    end
   else if c = #075 then
    begin
     envia_dato(PSER,$50);
     sign;
     write(' Giro de Mano + ');
     rest;
    end
   else if c = #45 then
    begin
     envia_dato(PSER,$53);
     sign;
     write(' Todos en Reversa ');
     rest;
    end
   else if c = #09 then
    begin
     envia_dato(PSER,$52);
     sign;
     write(' Todos hacia Adelante ');
     rest;
    end
  END;

PROCEDURE INI;
 BEGIN
  I:=0;DR:=0;DA:=0;SA:=0;SR:=0;
  BR:=0;BA:=0;CR:=0;CA:=0;h2:=0;m2:=0;
  MR:=0;MA:=0;RR:=0;HR:=0;h1:=0;m1:=0;
  HA:=0;TA:=0;TR:=0;AA:=0;
 END;

PROCEDURE ORIGEN;
 BEGIN
  RESET(mar);
      q:=0;
      INI;
      WHILE EOF(MAR) DO
       BEGIN
        READ(MAR,C);
        Q:=Q+1;
       END;

      Q:=Q-1;
      reset(MAR);
      PE:=#00;
      WHILE Q <> -1 DO
       BEGIN
        SEEK(MAR,Q);
        READ(MAR,C);
        IF PE= #32 THEN
         Q:=Q-1
        ELSE
         BEGIN
          IF KEYPRESSED THEN
           BEGIN
            PE:=READKEY;
            IF PE <> #32 THEN
             BEGIN
              INICIO(C);
              Q:=Q-1;
             END
            ELSE IF PE= #32 THEN
             Q:=q-1;
           END
          ELSE
           BEGIN
            INICIO(C);
            Q:=Q-1;
           END
         END
       END;
 END;

PROCEDURE CERO(I:INTEGER);
 BEGIN
  PE:=#00;
  IF DA >= DR THEN
   BEGIN
    DA:=DA-DR;
    DR:=0;
   END
  ELSE
   BEGIN
    DR:=DR-DA;
    DA:=0;
   END;
  IF AA >= RR THEN
   BEGIN
    AA:=AA-RR;
    RR:=0;
   END
  ELSE
   BEGIN
    RR:=RR-AA;
    AA:=0;
   END;
```

```
IF BA >= BR THEN
 BEGIN
  BA:=BA-BR;
  BR:=0;
 END
ELSE
 BEGIN
  BR:=BR-BA;
  BA:=0;
 END;
IF CA >= CR THEN
 BEGIN
  CA:=CA-CR;
  CR:=0;
 END
ELSE
 BEGIN
  CR:=CR-CA;
  CA:=0;
 END;
IF HA >= HR THEN
 BEGIN
  HA:=HA-HR;
  HR:=0;
 END
ELSE
 BEGIN
  HR:=HR-HA;
  HA:=0;
 END;
IF TA >= TR THEN
 BEGIN
  TA:=TA-TR;
  TR:=0;
 END
ELSE
 BEGIN
  TR:=TR-TA;
  TA:=0;
 END;
IF MA >= MR THEN
 BEGIN
  MA:=MA-MR;
  MR:=0;
 END
ELSE
 BEGIN
  MR:=MR-MA;
  MA:=0;
 END;
IF m1 >= h1 THEN
 BEGIN
  M1:=M1-h1;
  h1:=0;
 END
ELSE
 BEGIN
  h1:=h1-m1;
  m1:=0;
 END;
IF m2 >= h2 THEN
 BEGIN
  M2:=M2-h2;
  h2:=0;
 END
ELSE
 BEGIN
  h2:=h2-m2;
  m2:=0;
 END;
IF sa >= sr THEN
 BEGIN
  sa:=sa-sr;
  sr:=0;
 END
ELSE
 BEGIN
  sr:=sr-sa;
  sa:=0;
 END;
FOR I:=1 TO MR DO
 begin
  IF PE = #32 THEN
   I:=MR
  ELSE
   BEGIN
    envia_dato(PSER,$42);
    write(' ','MR');
    sign;
    write(' Mano en Reversa ');
    rest;
    IF KEYPRESSED THEN
     BEGIN
      PE:=READKEY;
      IF PE=#32 THEN
       I:=MR
     END;
   END;
 end;
FOR I:=1 TO MA DO
 begin
  IF PE = #32 THEN
   I:=MA
  ELSE
   begin
    envia_dato(PSER,$43);
    write(' ','MA');
    sign;
    write(' Mano hacia Adelante ');
    rest;
    IF KEYPRESSED THEN
     BEGIN
      PE:=READKEY;
      IF PE=#32 THEN
       I:=MA
     END;
   end;
 end;
FOR I:=1 TO h2 DO
 begin
  IF PE = #32 THEN
   I:=h2
  ELSE
   BEGIN
```

```
      envia_dato(PSER,$50);
      write(' ','G+');
      sign;
      write(' Giro de Mano + ');
      rest;
      IF KEYPRESSED THEN
       BEGIN
        PE:=READKEY;
        IF PE=#32 THEN
         I:=h2
       END;
     END;
  end;
FOR i:=1 TO m2 DO
  begin
   IF PE = #32 THEN
    I:=m2
   ELSE
    begin
     envia_dato(PSER,$51);
     write(' ','G-');
     sign;
     write(' Giro de Mano - ');
     rest;
     IF KEYPRESSED THEN
      BEGIN
       PE:=READKEY;
       IF PE=#32 THEN
        I:=m2
      END;
    end;
  end;
FOR i:=1 TO BR DO
  begin
   IF PE = #32 THEN
    I:=BR
   ELSE
    BEGIN
     envia_dato(PSER,$37);
     write(' ','BR');
     sign;
     write(' Base Sentido Antihorario ');
     rest;
     IF KEYPRESSED THEN
      BEGIN
       PE:=READKEY;
       IF PE=#32 THEN
        I:=BR;
      END;
    END;
  end;
FOR I:=1 TO CR DO
  begin
   IF PE = #32 THEN
    I:=CR
   ELSE
    BEGIN
     envia_dato(PSER,$38);
     write(' ','CR');
     sign;
     Write(' Codo en Reverse ');
     rest;
     IF KEYPRESSED THEN
      BEGIN
       PE:=READKEY;
       IF PE=#32 THEN
        I:=CR;
      END;
    END;
  end;
FOR I:=1 TO HR DO
  begin
   IF PE = #32 THEN
    I:=HR
   ELSE
    BEGIN
     envia_dato(PSER,$36);
     write(' ','HR');
     sign;
     write(' Hombro en Reversa ');
     rest;
     IF KEYPRESSED THEN
      BEGIN
       PE:=READKEY;
       IF PE=#32 THEN
        I:=HR;
      END;
    END;
  end;
FOR i:=1 TO BA DO
  begin
   IF PE = #32 THEN
    I:=BA
   ELSE
    BEGIN
     envia_dato(PSER,$34);
     write(' ','BA');
     sign;
     Write(' Base Sentido Horario ');
     rest;
     IF KEYPRESSED THEN
      BEGIN
       PE:=READKEY;
       IF PE=#32 THEN
        I:=BA;
      END;
    END;
  end;
FOR i:=1 TO CA DO
  begin
   IF PE = #32 THEN
    I:=CA
   ELSE
    BEGIN
     envia_dato(PSER,$35);
     write(' ','CA');
     sign;
     Write(' Codo hacia Adelante ');
     rest;
     IF KEYPRESSED THEN
      BEGIN
       PE:=READKEY;
       IF PE=#32 THEN
        I:=CA;
```

```
        END;
      END;
    end;
  FOR I:=1 TO HA DO
    begin
      IF PE = #32 THEN
        I:=HA
      ELSE
        BEGIN
          envia_dato(PSER,$39);
          write(' ','HA');
          sign;
          write(' Hombro hacia Adelante ');
          rest;
          IF KEYPRESSED THEN
            BEGIN
              PE:=READKEY;
              IF PE=#32 THEN
                I:=HA;
            END;
        END;
    end;
  FOR I:=1 TO AA DO
    begin
      IF PE = #32 THEN
        I:=AA
      ELSE
        BEGIN
          envia_dato(PSER,$46);
          write(' ','AA');
          sign;
          writeln('  Codo hacia Adelante  ');
          write(' Hombro hacia Adelante ');
          rest;
          IF KEYPRESSED THEN
            BEGIN
              PE:=READKEY;
              IF PE=#32 THEN
                I:=AA;
            END;
        END;
    end;
  FOR I:=1 TO RR DO
    begin
      IF PE = #32 THEN
        I:=RR
      ELSE
        BEGIN
          envia_dato(PSER,$33);
          write(' ','RR');
          sign;
          writeln('  Codo en Reversa  ');
          write(' Hombro en Reversa ');
          rest;
          IF KEYPRESSED THEN
            BEGIN
              PE:=READKEY;
              IF PE=#32 THEN
                I:=RR;
            END;
        END;
    end;
```

```
  FOR I:=1 TO DA DO
    BEGIN
      IF PE = #32 THEN
        I:=DA
      ELSE
        BEGIN
          ENVIA_DATO(PSER,$32);
          write(' ','DA');
          sign;
          writeln(' Base Sentido Horario ');
          write('  Codo hacia Adelante  ');
          rest;
          IF KEYPRESSED THEN
            BEGIN
              PE:=READKEY;
              IF PE=#32 THEN
                I:=DA;
            END;
        END;
    END;
  FOR I:=1 TO DR DO
    begin
      IF PE = #32 THEN
        I:=DR
      ELSE
        BEGIN
          envia_dato(PSER,$31);
          write(' ','DR');
          sign;
          writeln(' Base Sentido Antihorario ');
          write('     Codo en Reversa     ');
          rest;
          IF KEYPRESSED THEN
            BEGIN
              PE:=READKEY;
              IF PE=#32 THEN
                I:=DR;
            END;
        END;
    end;
  FOR I:=1 TO h1 DO
    begin
      IF PE = #32 THEN
        I:=h1
      ELSE
        BEGIN
          envia_dato(PSER,$47);
          write(' ','2R');
          sign;
          writeln(' Base Sentido Antihorario ');
          write('    Hombro en Reversa    ');
          rest;
          IF KEYPRESSED THEN
            BEGIN
              PE:=READKEY;
              IF PE=#32 THEN
                I:=h1
            END;
        END;
    end;
  FOR i:=1 TO m1 DO
    begin
```

```
    IF PE = #32 THEN
      i:=m1
    ELSE
      begin
        envia_dato(PSER,$49);
        write(' ','2A');
        sign;
        writeln(' Base Sentido Horario ');
        write(' Hombro hacia Adelante ');
        rest;
        IF KEYPRESSED THEN
          BEGIN
            PE:=READKEY;
            IF PE=#32 THEN
              i:=m1
          END;
      end;
  end;
FOR i:=1 TO TA DO
  begin
    IF PE = #32 THEN
      i:=TA
    ELSE
      BEGIN
        envia_dato(PSER,$41);
        write(' ','TA');
        sign;
        writeln(' Base Sentido Horario ');
        writeln(' Codo hacia Adelante ');
        write(' Hombro hacia Adelante ');
        rest;
        IF KEYPRESSED THEN
          BEGIN
            PE:=READKEY;
            IF PE=#32 THEN
              i:=TA;
          END;
      END;
  end;
FOR i:=1 TO TR DO
  begin
    IF PE = #32 THEN
      i:=TR
    ELSE
      BEGIN
        envia_dato(PSER,$40);
        write(' ','TR');
        sign;
        writeln(' Base Sentido Antihorario ');
        writeln('     Codo en Reversa      ');
        write('    Hombro en Reversa     ');
        rest;
        IF KEYPRESSED THEN
          BEGIN
            PE:=READKEY;
            IF PE=#32 THEN
              i:=TR;
          END;
      END;
  end;
FOR i:=1 TO sr DO
  begin
    IF PE = #32 THEN
      i:=sr
    ELSE
      BEGIN
        envia_dato(PSER,$53);
        write(' ','SR');
        sign;
        write(' Todos en Reversa ');
        rest;
        IF KEYPRESSED THEN
          BEGIN
            PE:=READKEY;
            IF PE=#32 THEN
              i:=sr
          END;
      END;
  end;
FOR i:=1 TO sa DO
  begin
    IF PE = #32 THEN
      i:=sa
    ELSE
      begin
        envia_dato(PSER,$52);
        write(' ','SA');
        sign;
        write(' Todos hacia Adelante ');
        rest;
        IF KEYPRESSED THEN
          BEGIN
            PE:=READKEY;
            IF PE=#32 THEN
              i:=sa
          END;
      end;
  end;
END;

FUNCTION EXISTE(FNAME:STRING):BOOLEAN;
  VAR f:FILE;
  BEGIN
    ASSIGN(F,FNAME);
    {$I-} RESET(F); {$I+}
    EXISTE:=(IORESULT=0);
  END;

procedure Trayectoria;
    begin
        AUX:='0';
        n:=0;
        S:=0;
        WRITELN;
        writeln('DESEA REGRESAR A LA POSICION
INICIAL Y/N ?');
        REPEAT
          AUX:=READKEY;
          AUX:=UPCASE(AUX);
          IF AUX='Y' THEN
            BEGIN
              WRITELN;
              writeln('TRAYECTORIAASEGUIR:LARGA"L"
o CORTA "C"');
```

```
            REPEAT
             BEGIN
              AUX:=READKEY;
              AUX:=UPCASE(AUX);
              IF AUX='C' THEN
               BEGIN
                S:=10;
                CERO(I);
               END
              ELSE IF AUX='L' THEN
               BEGIN
                ASSIGN(RUTINA1,NOMBRE);
                Q:=Q-1;
                S:=10;
                PE:=#00;
                RESET(RUTINA1);
                WHILE Q<>-1 DO
                 BEGIN
                  SEEK(RUTINA1,Q);
                  READ(RUTINA1,C);
                  IF PE = #32 THEN
                   Q:=q-1
                  ELSE
                   BEGIN
                    IF KEYPRESSED THEN
                     BEGIN
                      PE:=READKEY;
                      IF PE <> #32 THEN
                       BEGIN
                        INICIO(C);
                        Q:=Q-1;
                       END
                      else if PE=#32 then
                       Q:=q-1;
                     END
                    ELSE
                     BEGIN
                      INICIO(C);
                      Q:=Q-1;
                     END
                   END
                 END;
                WRITELN;
                CLOSE(RUTINA1);
                WRITELn;
               END;
              WRITELN;
             END
            UNTIL S=10;
            N:=90;
            Sign;
            write(' PUNTO DE REPOSO ');
            rest1;
           END
          ELSE IF AUX='N' THEN
           N:=90;
         UNTIL N=90;
         AUX:='0';
         INI;
         REWRITE(MAR);
end;
```

```
procedure PANTALLA;
 begin
  window(1,1,36,18);
  LOWVIDEO;
  TEXTBACKGROUND(15);
  CLRSCR;
  GOTOXY(10,1);
  TEXTATTR:=YELLOW+BLUE*16;
  WRITE(' MENU DE ACCIONES ');
  WINDOW(1,2,36,18);
  TEXTBACKGROUND(15);
  TEXTCOLOR(1);
  WRITELN(' F1 --> GIRAR BASE   (MOTOR 1)    ');
  WRITELN(' F2 --> MOVER CODO   (MOTOR 2)    ');
  WRITELN(' F3 --> MOVER HOMBRO (MOTOR 3)    ');
  WRITELN(' F4 --> MOVER TENAZA (MOTOR 4)    ');
  WRITELN(' F5 --> GIRAR MANO (MOTOR 5,6)    ');
  WRITELN(' F6 --> MOVER MANO (MOTOR 3,4)    ');
  WRITELN(' F7 --> MOVER BASE Y CODO         ');
  WRITELN(' F8 --> MOVER HOMBRO Y BASE       ');
  WRITELN(' F9 --> MOVER HOMBRO Y CODO       ');
  WRITELN(' F10 -> MOVER HOMBRO, BASE Y CODO ');
  WRITELN(' SHIFT-F1  REPETIR RUTINA APRENDIDA');
  WRITELN(' SHIFT-F2  EJECUTAR RUTINA GRABADA ');
  WRITELN('  L --> LIMPIAR ARCHIVO           ');
  WRITELN('  R --> RECIBIR DATOS DE LA SONDA  ');
  WRITELN('  I --> REGRESAR AL PUNTO DE INICIO');
  WRITELN(' ESC -> SALIR                     ');
  TEXTBACKGROUND(10);
  TEXTCOLOR(0);
  WRITE(' BARRA ');
  GOTOXY(12,17);

  WRITE(' PARO DE EMERGENCIA');
  TEXTCOLOR(WHITE);
  TEXTBACKGROUND(0);
  sign;
  write('       INMOVIL     ');
  rest1;
 END;

(*PROGRAMA PRINCIPAL*)

BEGIN
 Assign(F,'MOVIL.PCX');
 RESET(F);
 TAM:=FileSize(F);
 EXEC('viewer.exe','MOVIL.PCX');
 codigo:=DosExitCode;
 if ((codigo <> 1 AND 0) AND (TAM = 23719)) then
  begin
  WITH REGS DO
   BEGIN
    AH:=$10;
    AL:=01;
    BH:=11;
    INTR($10,REGS);
   END;
  textbackground(1);
  TEXTCOLOR(15);
  CLRSCR;
  WRITELN;WRITELN;
```

```
      WRITELN('
╔═══════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════
═══╗');
  WRITELN('          ||                    BIENVENIDO
        ||');
  WRITELN('          ||
  ||');
  WRITELN('              ||  SISTEMA DE CONTROL Y
APRENDIZAJE DE LA SONDA ROBOT  ||');
      WRITELN('
╚═══════════════════════════════════════════════════════════════
═══════════════════════════════════════════════════════════════
═══╝');
  WRITELN;WRITELN;WRITELN;
  WRITELN('          SELECCIONE EL PUERTO SERIAL AL
CUAL EL SISTEMA ESTA');
  WRITE('          CONECTADO (COM1 = 1 o COM2 = 2):
');
  C:=#00;
  SAL:=00;
  repeat
    aux:=readkey;
    IF AUX='1' THEN
     BEGIN
      PSER:=0;
      sal:=49;
      WRITELN;WRITELN;WRITELN;
      WRITELN('              SE HA SELECCIONADO EL
PUERTO SERIAL 1');
      WRITELN;
     END

    ELSE
     begin
      if aux='2' then
       BEGIN
        PSER:=1;
        sal:=49;
        WRITELN;WRITELN;WRITELN;
        WRITELN('              SE HA SELECCIONADO EL
PUERTO SERIAL 2');
        WRITELN;
       END;
     end;
  until sal=49;
  WRITELN;
  REPEAT
   repeat
    WRITE('          ESCRIBA EL RETARDO QUE DESEE
(ms): ');
    readln(retardo);
   until length(retardo) < >0;
   val(retardo,ms,error);
  UNTIL error=0;
  WITH REGS DO
   BEGIN
    AH:=$10;
    AL:=01;
    BH:=36;
    INTR($10,REGS);
   END;
```

```
  textbackground(0);
  CLRSCR;
  INICIALIZA_BAUDAJE(PSER);
  RESETEA_PTO_SERIAL(PSER);
  PANTALLA;
  C:=#00;
  WITH REGS DO
   BEGIN
    AH:=01;
    CH:=$20;   {APAGAR CURSOR}
    INTR($10,REGS);       {INTERRUPCON  PARA  EL
CURSOR}
   END;

  REPEAT
   C:=READKEY;
   C:=UPCASE(C);
  UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR (C=#61)
OR (C=#62) OR (C=#85) OR (C=#84) OR (C='I')
      OR (C=#63) OR (C=#64) OR (C=#65) OR (C=#66)
OR (C=#67) OR (C=#68) OR (C='L') OR (C='R'));

  repeat

    if C=#065 then
     begin
      WINDOW(2,8,36,8);
      HIGHVIDEO;
      TEXTBACKGROUND(1);
      WRITE('F7 --> MOVER BASE Y CODO ');
      WINDOW(1,1,36,17);
      LOWVIDEO;
      C:=#00;
      REPEAT
      IF KeyPressed THEN
       BEGIN
        C:=READKEY;
        C:=UPCASE(C);
        IF C=#072 THEN
         BEGIN
          envia_dato(PSER,$32);
          Dt:=#49;
          WRITE(MAR,D);
          sign:
          writeln('  Base Giro Horario  ');
          write(' Codo hacia Adelante ');
          rest1;
          DR:=DR+1;
         END
        else if c=#080 then
         begin
          envia_dato(PSER,$31);
          D:=#50;
          WRITE(MAR,D);
          sign;
          writeln('    Codo en Reversa    ');
          write(' Base Giro Antihorario ');
          rest1;
          DA:=DA+1;
         end
       END
      UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
```

```
(C = #61) OR (C = #62) OR (C = #85) OR (C = #84) OR
(C = 'I')
         OR (C = 'L') OR (C = 'R') OR (C = #63) OR
(C = #64) OR (C = #65) OR (C = #66) OR (C = #67) OR
(C = #68));
        PANTALLA;
       END {IF}
      ELSE if C = #066 then
       begin
        WINDOW(2,9,36,9);
        HIGHVIDEO;
        TEXTBACKGROUND(1);
        HIGHVIDEO;
        WRITE('F8 -> MOVER HOMBRO Y BASE');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          if c = #072 then
           begin
            envia_dato(PSER,$49);
            write(mar,D);
            sign;
            writeln('   Base Giro Horario   ');
            write(' Hombro hacia Adelante ');
            rest1;
            h1:=h1+1;
           end
          else if c = #080 then
           begin
            envia_dato(PSER,$47);
            write(mar,D);
            sign;
            writeln(' Base Giro Antihorario ');
            write('   Hombro en Reversa   ');
            rest1;
            m1:=m1+1;
           end
         END
        UNTIL ((C = #27) OR (C = #59) OR (C = #60) OR
(C = #61) OR (C = #62) OR (C = #63) OR (C = 'I')
         OR (C = 'L') OR (C = 'R') OR (C = #64) OR
(C = #65) OR (C = #66) OR (C = #67) OR (C = #68)) OR
(C = #85) OR (C = #84);
        PANTALLA;
       END {IF}
      else if C = #067 then
       begin
        WINDOW(2,10,36,10);
        HIGHVIDEO;
        TEXTBACKGROUND(1);
        WRITE('F9 -> MOVER HOMBRO Y CODO');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          IF C = #080 THEN
           begin
            envia_dato(PSER,$33);
            D:=#51;
            WRITE(MAR,D);
            sign;
            writeln('  Codo en Reversa   ');
            write(' Hombro en Reversa ');
            rest1;
            AA:=AA+1;
           end
          else if c = #072 then
           begin
            envia_dato(PSER,$46);
            D:='+';
            WRITE(MAR,D);
            sign;
            writeln('  Codo hacia Adelante  ');
            write(' Hombro hacia Adelante ');
            rest1;
            RR:=RR+1;
           end
         END
        UNTIL ((C = #27) OR (C = #59) OR (C = #60) OR
(C = #61) OR (C = #62) OR (C = #85) OR (C = #84) OR
(C = 'I')
         OR (C = 'L') OR (C = 'R') OR (C = #63) OR
(C = #64) OR (C = #65) OR (C = #66) OR (C = #67) OR
(C = #68));
        PANTALLA;
       END
      ELSE if C = #068 then
       begin
        WINDOW(2,11,36,11);
        HIGHVIDEO;
        TEXTBACKGROUND(1);
        WRITE('F10 -> MOVER HOMBRO, BASE Y
CODO');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          if c = #072 then
           begin
            envia_dato(PSER,$41);
            D:='0';
            WRITE(MAR,D);
            sign;
            writeln('  Base Giro Horario   ');
            writeln('  Codo hacia Adelante  ');
            write(' Hombro hacia Adelante ');
            rest1;
            TR:=TR+1;
           end
          else if c = #080 then
           begin
```

```
            envia_dato(PSER,$40);
            D:=#46;
            WRITE(MAR,D);
            sign;
            writeln(' Base Giro Antihorario ');
            writeln('   Codo en Reversa   ');
            write('  Hombro en Reversa  ');
            rest1;
            TA:=TA+1;
           end
         END
        UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
         OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
        PANTALLA;
       END
      ELSE if C=#059 then
       begin
        WINDOW(2,2,36,2);
        HIGHVIDEO;
         TEXTBACKGROUND(1);
        WRITE('F1 --> GIRAR BASE   (MOTOR 1)');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          if c=#077 then
           begin
            envia_dato(PSER,$37);
            D:=#52;
            WRITE(MAR,D);
            sign;
            write(' Base Sentido Antihorario ');
            rest1;
            BA:=BA+1;
           end
          ELSE if c=#075 then
           begin
            envia_dato(PSER,$34);
            D:='7';
            WRITE(MAR,D);
            sign;
            write('  Base Giro Horario  ');
            rest1;
            BR:=BR+1;
           end
         END
        UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
         OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
        PANTALLA;
       END
      ELSE if C=#060 then
       begin
        WINDOW(2,3,36,3);
        HIGHVIDEO;
         TEXTBACKGROUND(1);
        WRITE('F2 --> MOVER CODO   (MOTOR 2)');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          if c=#080 then
           begin
            envia_dato(PSER,$38);
            D:=#53;
            WRITE(MAR,D);
            sign;
            write(' Codo en Reversa ');
            rest1;
            CA:=CA+1;
           end
          else if c=#072 then
           begin
            envia_dato(PSER,$35);
            D:='8';
            WRITE(MAR,D);
            sign;
            write(' Codo hacia Adelante ');
            rest1;
            CR:=CR+1;
           end
         END
        UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
         OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
        PANTALLA;
       END
      ELSE if C=#061 then
       begin
        WINDOW(2,4,36,4);
        HIGHVIDEO;
         TEXTBACKGROUND(1);
        WRITE('F3 --> MOVER HOMBRO (MOTOR 3)');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          if c=#080 then
           begin
            envia_dato(PSER,$36);
            D:=#54;
            WRITE(MAR,D);
```

```
            sign;
            write(' Hombro en Reversa ');
            rest1;
            HA:=HA+1;
           end
          else if c=#072 then
           begin
            envia_dato(PSER,$39);
            D:='9';
            WRITE(MAR,D);
            sign;
            write(' Hombro hacia Adelante ');
            rest1;
            HR:=HR+1;
           end
         END
        UNTIL ((C=#27) OR (C='9') OR (C='0') OR
(C='1') OR (C='2') OR (C='5') OR (C='4') OR
(C='I')
         OR (C='L') OR (C='R') OR (C='3') OR
(C='4') OR (C='5') OR (C='6') OR (C='7') OR
(C='8'));
        PANTALLA;
       END
      ELSE if C='2' then
       begin
        WINDOW(2,5,36,5);
        HIGHVIDEO;
         TEXTBACKGROUND(1);
        WRITE('F4 --> MOVER TENAZA (MOTOR 4)');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          if c=#072 then
           begin
            envia_dato(PSER,$42);
            D:='4';
            WRITE(MAR,D);
            sign;
            write(' Abrir Mano ');
            rest1;
            MA:=MA+1;
           end
          else if c=#080 then
           begin
            envia_dato(PSER,$43);
            D:='7';
            WRITE(MAR,D);
            sign;
            write(' Cerrar Mano ');
            rest1;
            MR:=MR+1;
           end
         END
        UNTIL ((C=#27) OR (C='9') OR (C='0') OR
(C='1') OR (C='2') OR (C='5') OR (C='4') OR
(C='I')
         OR (C='L') OR (C='R') OR (C='3') OR
(C='4') OR (C='5') OR (C='6') OR (C='7') OR
(C='8'));
        PANTALLA;
       END
      ELSE if C='3' then
       begin
        WINDOW(2,6,36,6);
        HIGHVIDEO;
         TEXTBACKGROUND(1);
        WRITE('F5 --> GIRAR MANO (MOTOR 5,6)');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          if c=#075 then
           begin
            envia_dato(PSER,$51);
            write(mar,D);
            sign;
            write('  Giro de Mano - ');
            rest1;
            h2:=h2+1;
           end
          else if c=#077 then
           begin
            envia_dato(PSER,$50);
            write(mar,D);
            sign;
            write('  Giro de Mano + ');
            rest1;
            m2:=m2+1;
           end
         END
        UNTIL ((C=#27) OR (C='9') OR (C='0') OR
(C='1') OR (C='2') OR (C='5') OR (C='4') OR
(C='I')
         OR (C='L') OR (C='R') OR (C='3') OR
(C='4') OR (C='5') OR (C='6') OR (C='7') OR
(C='8'));
        PANTALLA;
       END;
      if C='4' then
       begin
        WINDOW(2,7,36,7);
        HIGHVIDEO;
         TEXTBACKGROUND(1);
        WRITE('F6 --> MOVER MANO (MOTOR 3,4)');
        WINDOW(1,1,36,17);
        LOWVIDEO;
        C:=#00;
        REPEAT
        IF KeyPressed THEN
         BEGIN
          C:=READKEY;
          C:=UPCASE(C);
          if c=#72 then
           begin
```

```
            envia_dato(PSER,$52);
            write(mar,D);
            sign;
            write(' Mano hacia Adelante ');
            rest1;
            sr:=sr+1;
           end
          else if c=#80 then
           begin
            envia_dato(PSER,$53);
            write(mar,D);
            sign;
            write(' Mano en Reversa ');
            rest1;
            sa:=sa+1;
           end
         END
        UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
        OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
        PANTALLA;
       END
      else if C=#084 then
       begin
        WINDOW(38,1,80,24);
        CLRSCR;
        TEXTBACKGROUND(10);
        CLRSCR;
        HIGHVIDEO;
                        W R I T E L N ( '
═══════════════════════════════════
═══════════');
        WRITELN('      ▌      EJECUTANDO  RUTINA
APRENDIDA   ▌');
                        W R I T E L N ( '
═══════════════════════════════════
═══════════');
        WRITELN;
        Q:=Q-1;
        WRITELN('ESPERE UN MOMENTO...');
        ORIGEN;
        WRITELN;
        WRITELN('PUNTO DE INICIO');
        RESET(MAR);
        pe:=#00;
        Q:=0;
        WRITELn;
        writeln('EJECUTANDO SECUENCIA');
        WHILE EOF(MAR) DO
         BEGIN
          READ(MAR,C);
          EJECUTA;
         END;
        WRITELN;
        N:=0;

        writeln(' DESEA GUARDAR LA RUTINA ?');
        repeat
         aux:=readkey;
         AUX:=UPCASE(AUX);
         if aux='Y' then
          begin
           WRITELN;
           writeln;
           S:=0;
           REPEAT
            writeln('ESCRIBA EL NOMBRE DEL
ARCHIVO');
            REPEAT
             READLN(NOMBRE);
            until length(nombre) <>0;
            ST:='.SON';
            NOMBRE:=CONCAT(NOMBRE,ST);
            if existe(nombre) then
             writeln('ESTE ARCHIVO YA EXISTE')
            else
             begin
              ASSIGN(ARCH,'RUTINA.SON');
              RENAME(ARCH,NOMBRE);
              S:=10;
             end;
           WRITELN;
          UNTIL S=10;
          n:=90;
          ASSIGN(MAR,'RUTINA.SON');
         END
        else if aux='N' then
         BEGIN
          n:=90;
          NOMBRE:='RUTINA.SON';
         END;
       until n=90;
       trayectoria;
       WINDOW(38,1,80,24);
       TEXTBACKGROUND(0);
       CLRSCR;
       C:=#00;
         REPEAT
          C:=READKEY;
          C:=UPCASE(C);
         UNTIL (C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
        OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
       end


     ELSE IF C=#085 THEN
      BEGIN
       b:=0;
       WINDOW(38,1,80,24);
       CLRSCR;
       TEXTBACKGROUND(10);
       CLRSCR;
       HIGHVIDEO;
       WRITELN;
       WRITELN(' ESCOJA EL NUMERO DE LA RUTINA
A EJECUTAR');
       CONT:=1;
```

```
TEXTBACKGROUND(9);
GOTOXY(2,4);
WRITELN(' 0  > SALIR ');
TEXTBACKGROUND(10);
FINDFIRST('*.SON',0,SSR);
WHILE DOSERROR = 0 DO
 BEGIN
  WITH SSR DO
   WRITELN('  ',CONT,'--> ', NAME);
  FINDNEXT(SSR);
  WITH SSR DO
   BEGIN
    IF NAME = 'RUTINA.SON' THEN
     FINDNEXT(SSR);
    CONT:=CONT+1;
   END
 END;
 WRITELN;
REPEAT
 write('  OPCION A EJECUTAR:   #');
 repeat
  readln(opcion);
 until length(opcion) <>0;
 val(opcion,b,e);
 if e<>0 then
  b:=-1
UNTIL ((B>=0) AND (b<CONT));
CONT:=1;
WRITELn;
IF B<>0 THEN
 BEGIN
  FINDFIRST('*.SON',0,SSR);
  WHILE CONT<>b DO
   BEGIN
    FINDNEXT(SSR);
    WITH SSR DO
     BEGIN
      IF NAME = 'RUTINA.SON' THEN
       FINDNEXT(SSR);
      CONT:=CONT+1;
     END
   END;
  WITH SSR DO
   NOMBRE:=NAME;
  WRITELN('EJECUTANDO ',NOMBRE);
  IF EXISTE(NOMBRE) THEN
   BEGIN
    ASSIGN(RUTINA2,NOMBRE);
    RESET(RUTINA2);
    rewrite(mar);
    WRITELN;
    PE:=#00;
    O:=0;
    INI;
    WHILE EOF(RUTINA2) DO
     BEGIN
      READ(RUTINA2,C);
      EJECUTA;
     END;
    WRITELN;
    WRITELN;
    WRITE('SE HA TERMINADO DE EJECUTAR
EL ARCHIVO    ',NOMBRE);
    WRITELN('');
    CLOSE(RUTINA2);
    TRAYECTORIA;
   END
  ELSE WRITELN('EL ARCHIVO NO EXISTE');
 END;
WINDOW(38,1,80,24);
TEXTBACKGROUND(0);
CLRSCR;
C:=#00;
REPEAT
 C:=READKEY;
 C:=UPCASE(C);
UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
      OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
 END;

if c='L' then
 begin
  rewrite(mar);
  INI;
  C:=#00;
   REPEAT
    C:=READKEY;
    C:=UPCASE(C);
   UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
      OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
 end;
if c='I' then
 begin
  ORIGEN;
  Sign;
  writeln('  PUNTO DE INICIO  ');
  rest1;
  rewrite(mar);
  INI;
  C:=#00;
   REPEAT
    C:=READKEY;
    C:=UPCASE(C);
   UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
      OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
 end;

if c='R' then
 begin
  WINDOW(1,1,80,25);
  CLRSCR;
  WRITE('ESPERE UN MOMENTO....');
```

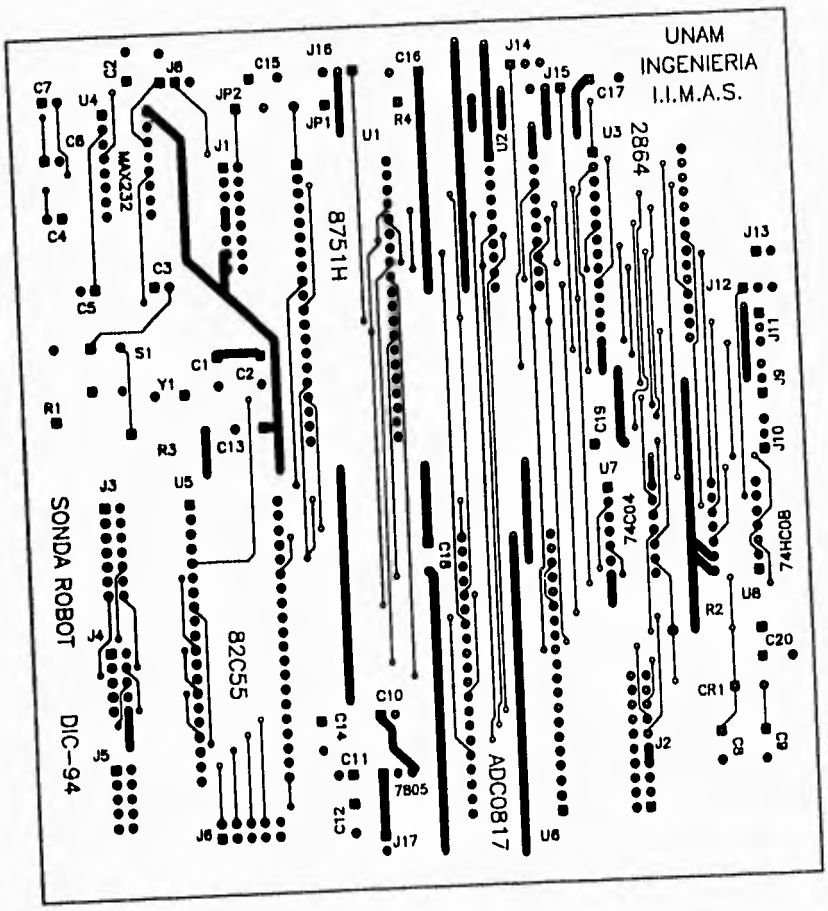
```
envia_dato(PSER,$78);
SAL:=00;
ASSIGN(RUTINA2,'DATOS.DAC');
REWRITE(RUTINA2);
RECIBE_DATO(PSER);
RECIBE_DATO(PSER);
conhex(num);
RECIBE_DATO(PSER);
conhex(num);
CLOSE(RUTINA2);
ASSIGN(RUTINA2,'DATOS.DAC');
total:=' ';
RESET(RUTINA2);
WHILE EOF(RUTINA2) do
 begin
  READ(RUTINA2,C);
  DECIMAL(C);
 end;
writeLN('total:',total);
CLOSE(RUTINA2);
val(total,cont,error);
if cont=0 then
else
 begin
  WRITE('cont:',CONT);
  if cont mod 7 = 0 then
  else
   begin
    rec:=cont/7;
    rec:=int(rec);
    rec:=rec*7;
   end;
  SAL:=00;
  clrscr;
  ASSIGN(RUTINA2,'OATOS.DAC');
  REWRITE(RUTINA2);
  REPEAT
   RECIBE_DATO(PSER);
   conhex(num);
   write(sal);
   SAL:=sal+1;
  UNTIL SAL=rec;
  CLOSE(RUTINA2);
  SAL:=00;
  CLRSCR;
  write(rec);

  RENGLON:=0;
  window(1,1,80,22);
   WRITELN('TIEMPO    |    DISTANCIA    |
ORIENTACION | TEMPERATURA | HUMEDAD | GAMA
| BETA ');
  WRITELN(' (seg) |    (m)    |    (m)    |
(°C)    |  (%)   | (Rad) | (Rad)');
  WRITELN('──────────┼─────────────
────┼──────────────┼──────────
─────┼───────────┼─────────
──');
  ASSIGN(RUTINA2,'DATOS.DAC');
  RESET(RUTINA2);
  REPEAT
   WRITE('  ');
        OCHO;
        WRITE('   | ');
        WRITE('   ');
        OCHO;
        WRITE('    | ');
        WRITE('    ');
        OCHO;
        WRITE('     | ');
        WRITE('    ');
        OCHO;
        WRITE('     | ');
        WRITE('   ');
        OCHO;
        WRITE('   | ');
        WRITE('  ');
        OCHO;
        WRITE('  | ');
        WRITE('  ');
        OCHO;
        WRITEln;
        RENGLON:=RENGLON+1;
        sal:=sal+1;
        IF RENGLON=20 THEN
         BEGIN
          aux:=#00;
          writeln('                    PRESIONAR
<RETURN> PARA AVANZAR');
          repeat
           aux:=readkey;
          until aux=#13;
          clrscr;
          WRITELN('TIEMPO    |    DISTANCIA    |
ORIENTACION | TEMPERATURA | HUMEDAD | GAMA
| BETA ');
          WRITELN(' (seg) |    (m)    |    (m)    |
(°C)    |  (%)   | (Rad) | (Rad)');
WRITELN('──────────┼──────────────┼──
─────────┼──────────────────┼──
──────────┼───────────┼──────────');
         end;
       UNTIL SAL=rec;
       CLOSE(RUTINA2);
      end;
     C:=#00;
     REPEAT
      C:=READKEY;
      C:=UPCASE(C);
     UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61) OR (C=#62) OR (C=#85) OR (C=#84) OR
(C='I')
      OR (C='L') OR (C='R') OR (C=#63) OR
(C=#64) OR (C=#65) OR (C=#66) OR (C=#67) OR
(C=#68));
     CLRSCR;
     PANTALLA;
    end;
   IF C=#27 THEN
    BEGIN
     AUX:=#00;
     window(4,20,35,25);
     TEXTBACKGROUND(12);
```

```
          CLRSCR;
          TEXTCOLOR(14);
          WRITELN;WRITELN;
          WRITELN('    TERMINAR EJECUCION ');
          WRITELN('   DEL PROGRAMA (Y/N) ?');
          SOUND(250);
          DELAY(500);
          NOSOUND;
          AUX:=READKEY;
          AUX:=UPCASE(AUX);
          IF AUX ='N' THEN
            BEGIN
              PANTALLA;
              Sign;
              write(' INMOVIL  ');
              rest1;
              C:=#00;
              REPEAT
                C:=READKEY;
                C:=UPCASE(C);
              UNTIL ((C=#27) OR (C=#59) OR (C=#60) OR
(C=#61)  OR  (C=#62)  OR  (C=#85)  OR  (C=#84)  OR
(C='I')
              OR  (C='L')  OR  (C='R')  OR  (C=#63)  OR
(C=#64)  OR  (C=#65)  OR  (C=#66)  OR  (C=#67)  OR
(C=#68));
            END;
        END
    until C=#27;
    WRITELN;
    WRITELN;
    CLOSE(MAR);
    WINDOW(1,22,60,25);
    CLRSCR;
    WRITELN ('DATOS ENVIADOS');
    WRITELN;
    WINDOW(1,1,60,25);
    textbackground(0);
    textcolor(15);
     LOWVIDEO;
    CLRSCR;
    WRITE('         OPRIMA TECLA DE <RETURN> PARA
CONTINUAR');
    END
    ELSE
     WRITELN('UNO O MAS ARCHIVOS NO FUERON
ENCONTRADOS');
    WITH REGS DO
     BEGIN
       AH:=$10;
       AL:=01;
       BH:=00;
       INTR($10,REGS);
     END;
END.
```

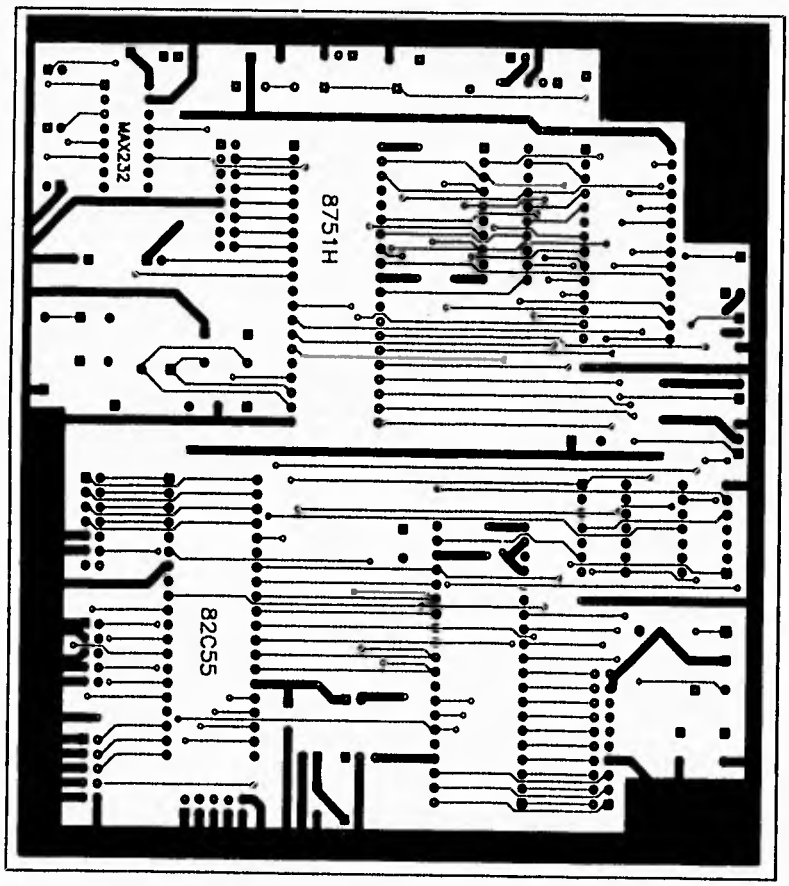
APÉNDICE C

TARJETA PRINCIPAL

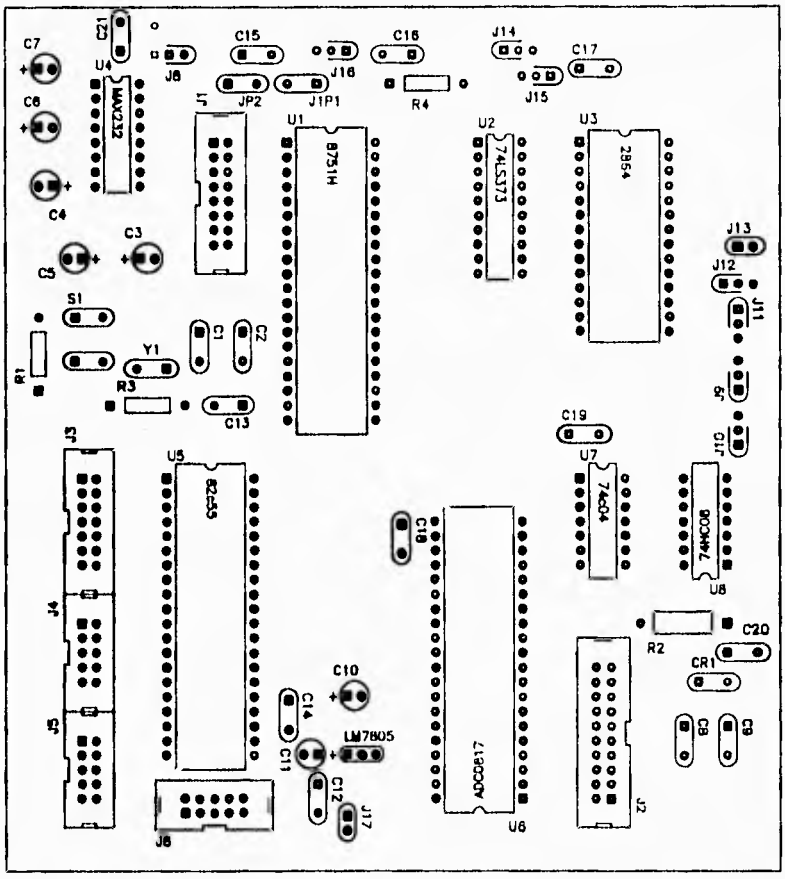
Apéndice C. Tarjeta Principal (Cara Superior).



Apéndice C. Tarjeta Principal (Cara Inferior).



Apéndice C. Tarjeta Principal (Ubicación de Componentes).



Lista de Componentes:

U1: 8751H
U2: 74LS373
U3: 2864
U4: MAX232
U5: 82C55
U6: ADC0817
U7: 74C04
U8: 74HC08

R1: 8.2 kilo ohms 1/2 Watt
R2: 1 Mega ohmm 1/2 Watt
R3: 100 ohms 1/2 Watt
R4: 2.7 kilo ohms 1/2 Watt

C1 y C2: 27 pF (cerámico)
C3 a C7: 10 micro Faradios, 16 volts (electrolítico)
C8 y C9: 470 pF (cerámico)
C10: 0.1 micro Faradios, 16 volts (Tantalio)
C11: 1 micro Faradio, 35 volts (Tantalio)
C12: 0.33 micro Faradios (cerámico)
C13 a C21: 0.1 micro Faradios (cerámico)

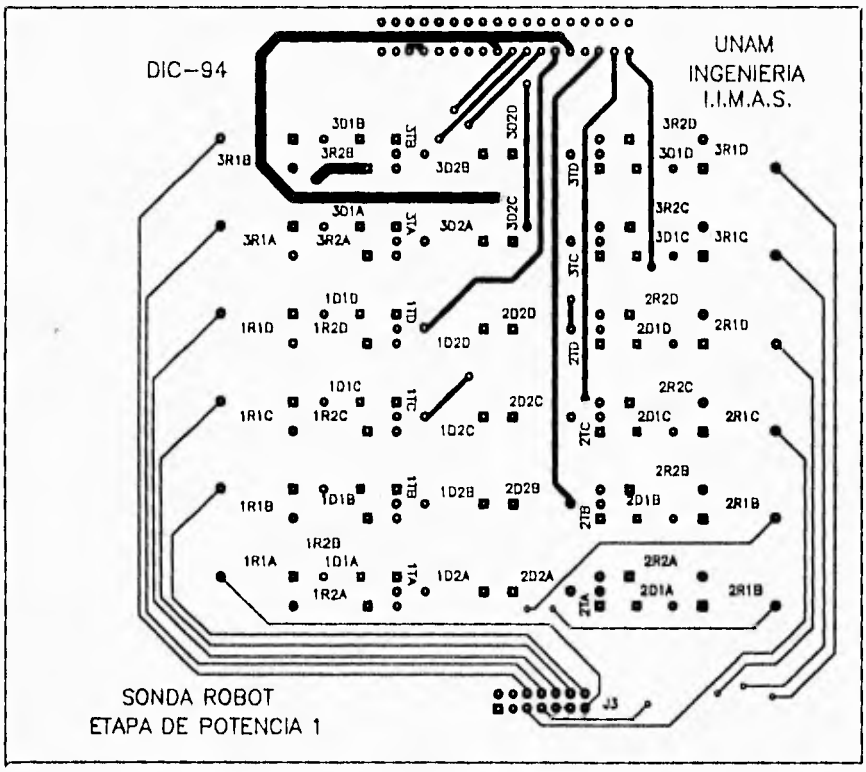
Y1: Cristal de cuarzo a 6 MHz
CR1: Oscilador a 640 kHz

J1: Conector Cable Plano FCN 16
J2: Conector Cable Plano FCN 20
J3: Conector Cable Plano FCN 14
J4: Conector Cable Plano FCN 10
J5: Conector Cable Plano FCN 10
J6: Conector Cable Plano FCN 10
J8 a J12: Poste Vertical 3 Posiciones
J13 y J17: Poste Vertical 2 Posiciones
J14 a J16: Poste Vertical 3 Posiciones

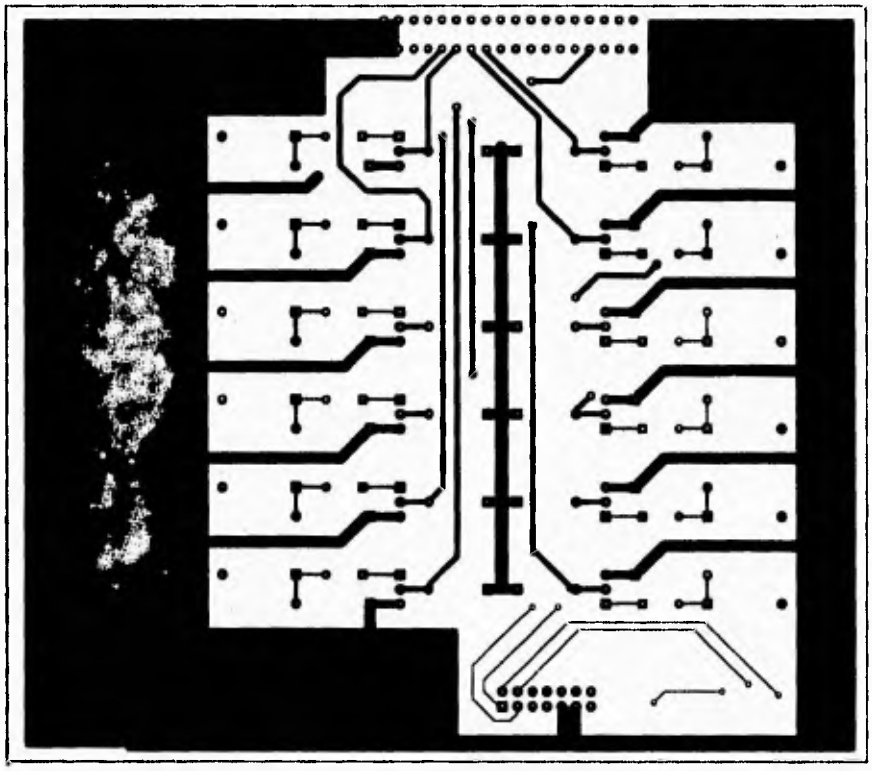
APÉNDICE D

ETAPA DE POTENCIA 1

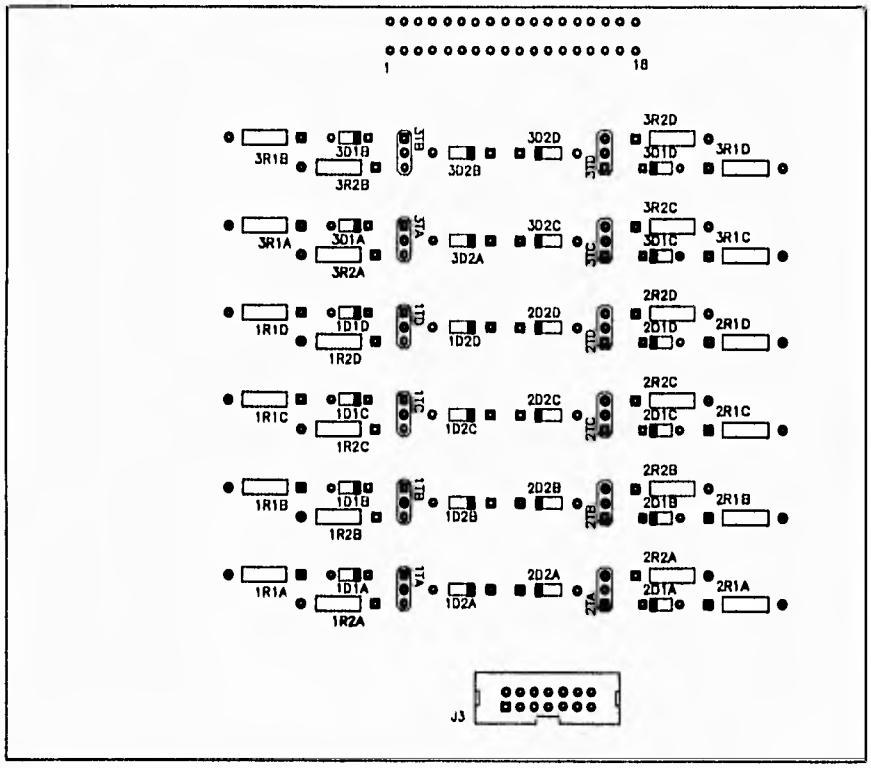
Apéndice D. Etapa de Potencia 1 (Cara Superior).



Apéndice D. Etapa de Potencia 1 (Cara Inferior).



Apéndice D. Etapa de Potencia 1 (Ubicación de Componentes).



Lista de Componentes:

1R1A a 1R1D: 1 kilo ohm 1/2 watt
1R2A a 1R2D: 100 kilo ohms 1/2 watt
1D1A a 1D1D: 1N4148
1D2A a 1D2D: 1N4007
1TA a 1TD: TIP 120

2R1A a 2R1D: 1 kilo ohm 1/2 watt
2R2A a 2R2D: 100 kilo ohms 1/2 watt
2D1A a 2D1D: 1N4148
2D2A a 2D2D: 1N4007
2TA a 2TD: TIP 120

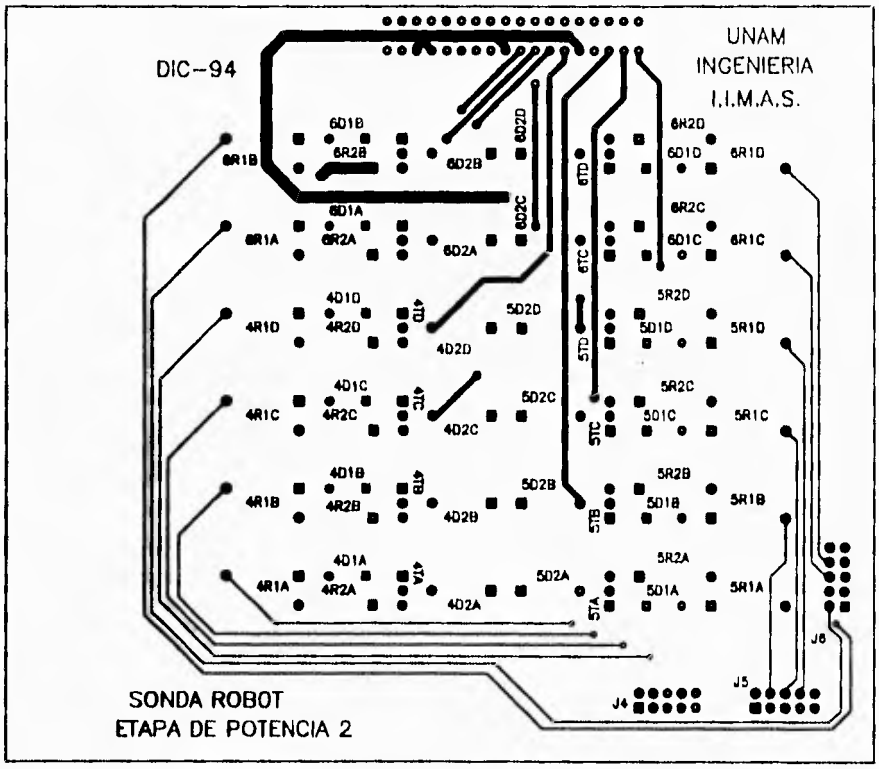
3R1A a 3R1D: 1 kilo ohm 1/2 watt
3R2A a 3R2D: 100 kilo ohms 1/2 watt
3D1A a 3D1D: 1N4148
3D2A a 3D2D: 1N4007
3TA a 3TD: TIP 120

J3: Conector Cable Plano FCN 14

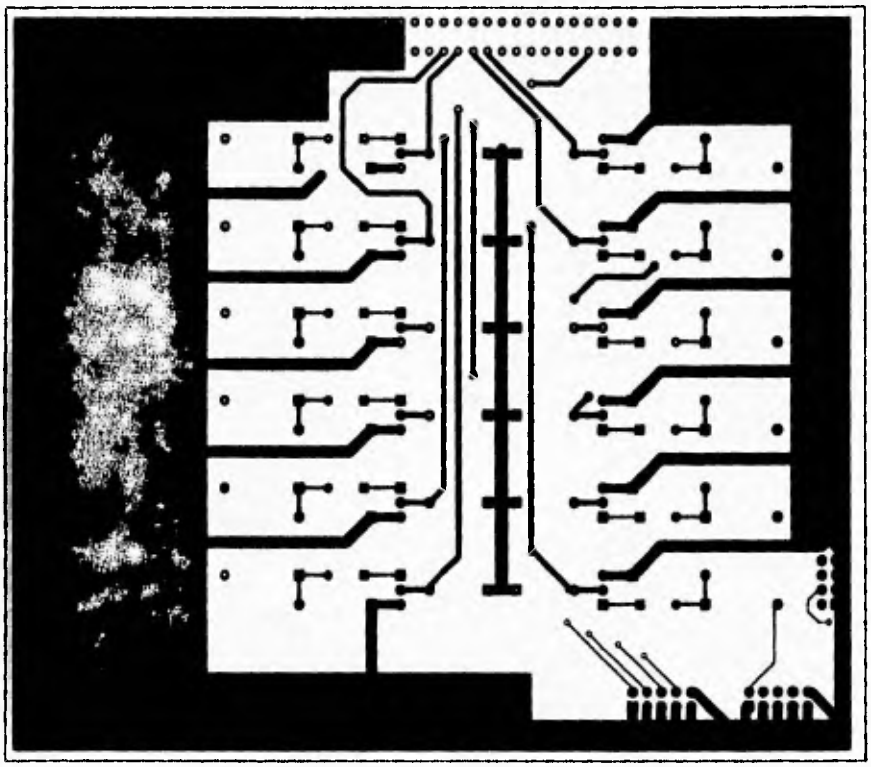
APÉNDICE E

ETAPA DE POTENCIA 2

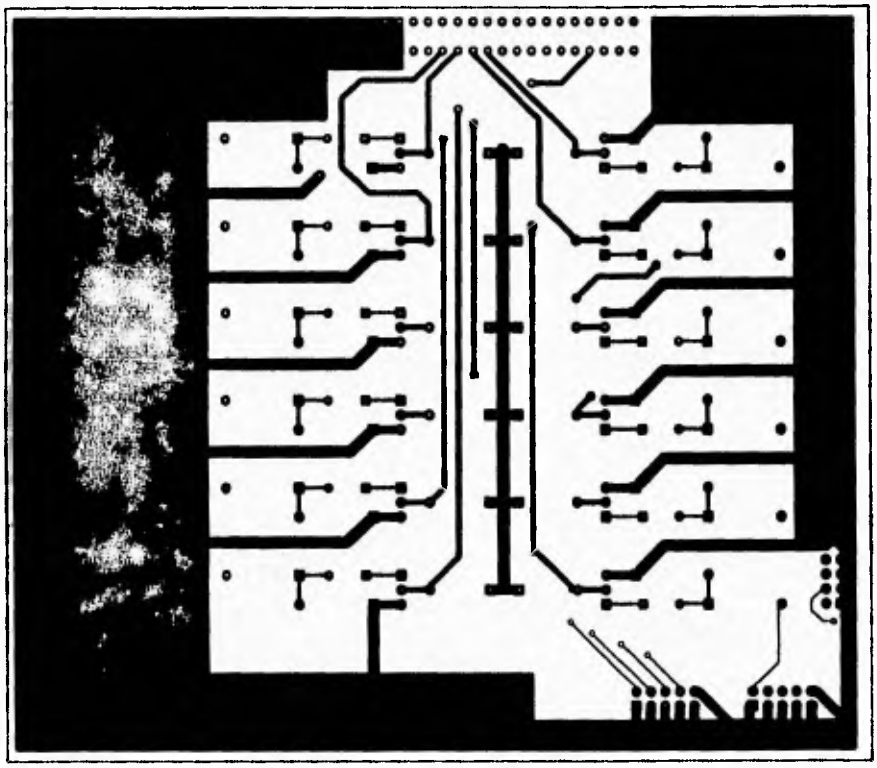
Apéndice E. Etapa de Potencia 2 (Cara Superior).



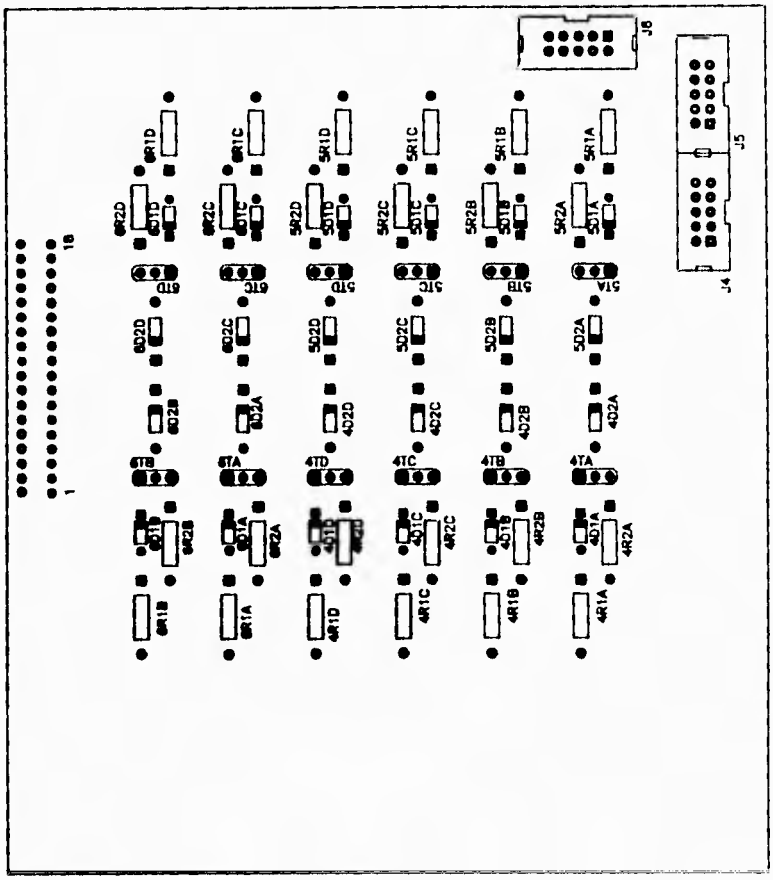
Apéndice E. Etapa de Potencia 2 (Cara Inferior).



Apéndice E. Etapa de Potencia 2 (Cara Inferior).



Apéndice E. Etapa de Potencia 2 (Ubicación de Componentes).



Lista de Componentes:

4R1A a 4R1D: 1 kilo ohm 1/2 watt
4R2A a 4R2D: 100 kilo ohms 1/2 watt
4D1A a 4D1D: 1N4148
4D2A a 4D2D: 1N4007
4TA a 4TD: TIP 120

5R1A a 5R1D: 1 kilo ohm 1/2 watt
5R2A a 5R2D: 100 kilo ohms 1/2 watt
5D1A a 5D1D: 1N4148
5D2A a 5D2D: 1N4007
5TA a 5TD: TIP 120

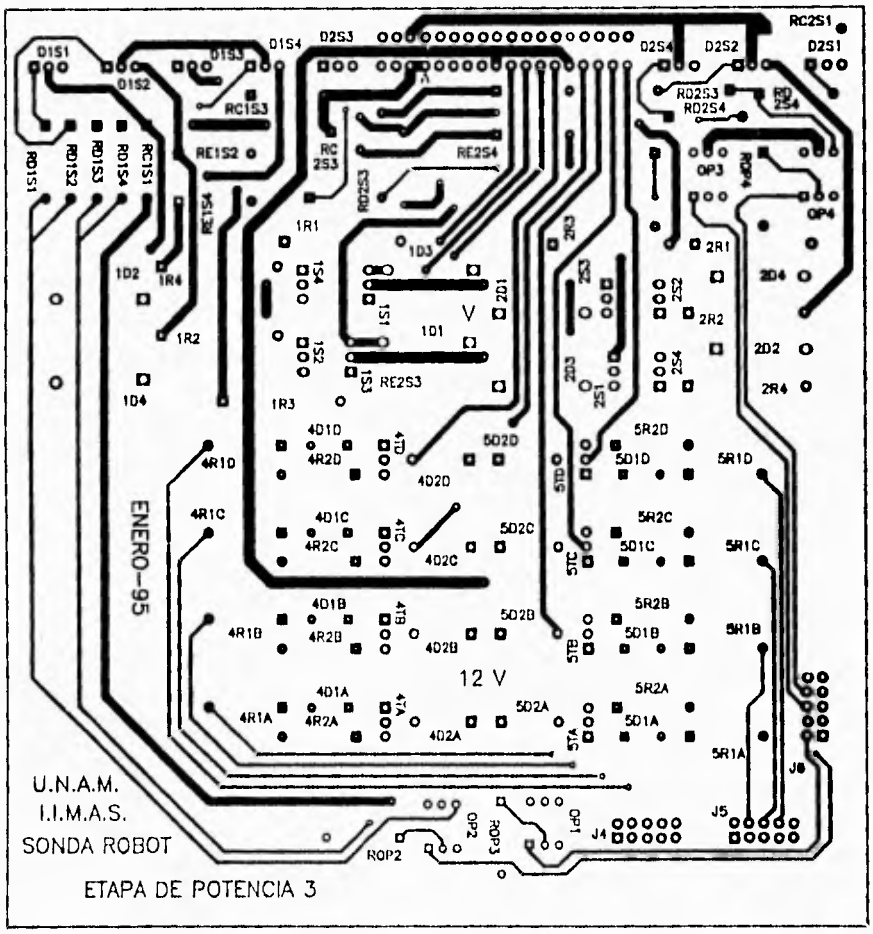
6R1A a 6R1D: 1 kilo ohm 1/2 watt
6R2A a 6R2D: 100 kilo ohms 1/2 watt
6D1A a 6D1D: 1N4148
6D2A a 6D2D: 1N4007
6TA a 6TD: TIP 120

J4 a J6: Conector Cable Plano FCN 10

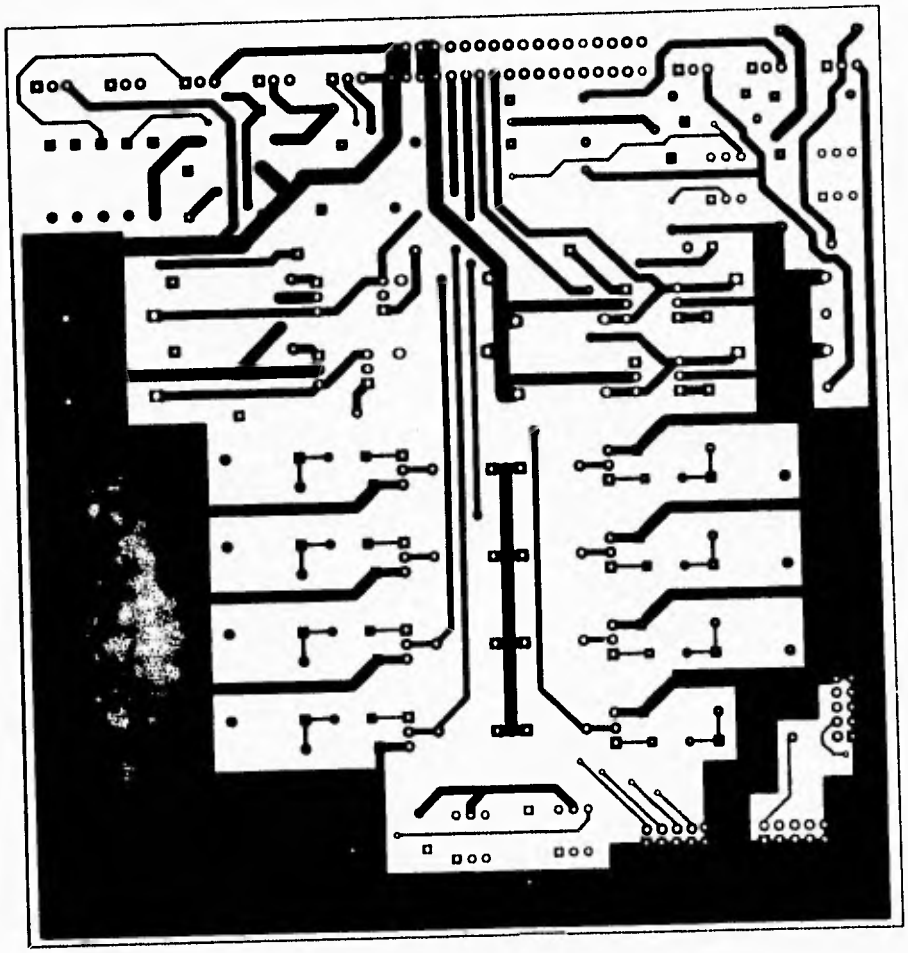
APÉNDICE F

ETAPA DE POTENCIA 3

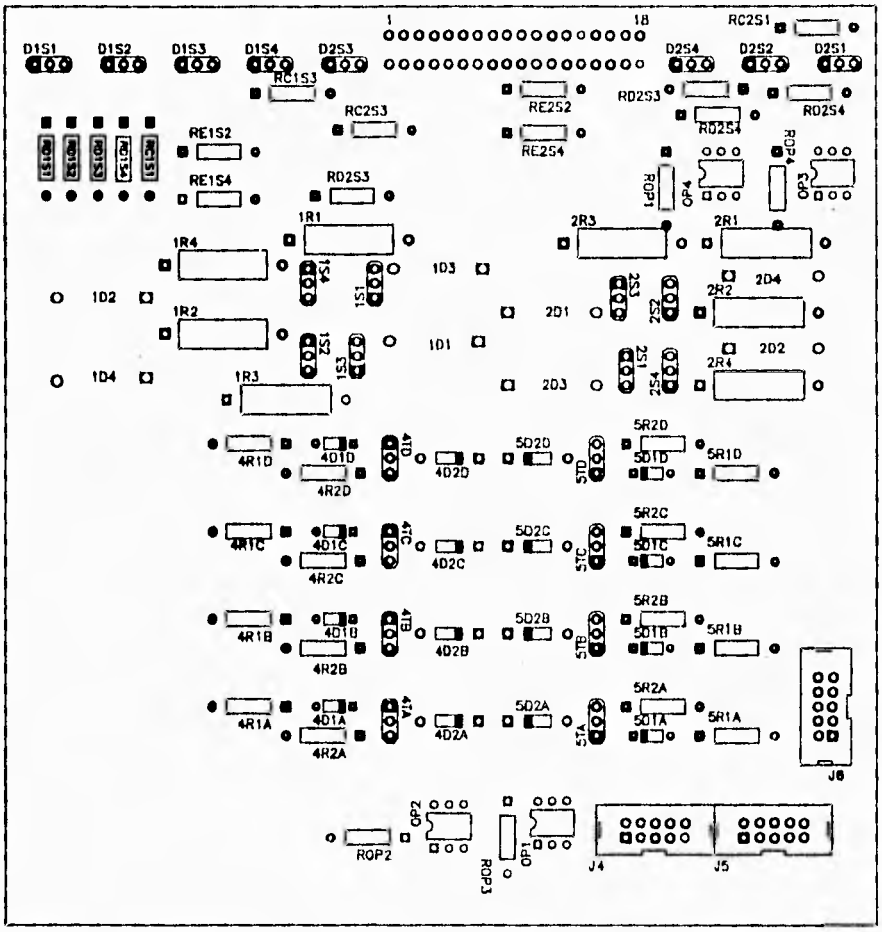
Apéndice F. Etapa de Potencia 3 (Cara Superior).



Apéndice F. Etapa de Potencia 3 (Cara Inferior).



Apéndice F. Etapa de Potencia 3 (Ubicación de Componentes).



Lista de Componentes:

4R1A a 4R1D: 1 kilo ohm 1/2 watt
4R2A a 4R2D: 100 kilo ohms 1/2 watt
4D1A a 4D1D: 1N4148
4D2A a 4D2D: 1N4007
4TA a 4TD: TIP 120

5R1A a 5R1D: 1 kilo ohm 1/2 watt
5R2A a 5R2D: 100 kilo ohms 1/2 watt
5D1A a 5D1D: 1N4148
5D2A a 5D2D: 1N4007
5TA a 5TD: TIP 120

J4 a J6: Conector Cable Plano FCN 10

D1S1 a D1S4 y D2S1 a D2S4: TIP 110

ROP1 a ROP4: 1.8 kilo ohms, 1/2 Watt

OP1 a OP4: 4N37

1D1 a 1D4 y 2D1 y 2D4: 1N5473

1R1 a 1R4: 100 ohms, 2 Watts

2R1 a 2R4: 100 ohms, 2 Watts

RC1S1, RC1S3, RC2S1 y RC2S3: 2.4 kilo ohms, 1/2 Watt

RE1S2, RE1S4, RE2S2 y RE2S4: 24 kilo ohms, 1/2 Watt

RD1S1 a RD1S4: 1.8 kilo ohms, 1/2 Watt

RD2S1 a RD2S4: 1.8 kilo ohms, 1/2 Watt

1S1, 1S3, 2S1 y 2S3: MJ2955

1S2, 1S4, 2S2 y 2S4: 2N3055

BIBLIOGRAFÍA

BIBLIOGRAFÍA

[1] 8 bits Embedded Controllers. Intel 1990

[2] Elementos de Robótica
P. Coiffer y M. Chirouze
Gustavo Gili (Colección Ciencia Electrónica) 1986

[3] Robótica Industrial: Tecnología, Programación y Aplicaciones
Groover, Weiss, Nagel y Odrey
Mc Graw-Hill, 1989

[4] Robótica: Control, Detección, Visión e Inteligencia.
Fu, González y Lee
Mc. Graw-Hill 1990

[5] Advanced MS-DOS
Ray Duncan
Microsoft Presentation 1988

[6] Mastering Turbo Pascal 6
Tom Swam
Hayden Books 1991 Cuarta Edición

[7] Electric Motors and Drivers
Austin Hughes
Newnes BH (Butterworth-Heinemann) 1990

[8] Máquinas Eléctricas
Fitzgerald, Kingsley y Umans
McGraw-Hill 1992 Quinta Edición

[9] Data Adquisition Handbook
National Semiconductor 1978

[10] The TTL Logic Data Book
Texas Instruments 1988

[11] Bipolar Power Transistor Data. Motorola 1989

[12] Electronics Now, Octubre 1993

[13] European Conference on Radiation and Its Effects on Devices and Systems. Montpelier, Francia. 1991